Institut für Informationssicherheit

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Beschreibung und Implementierung des 5G-AKA-Protokolls

Raphael Brösamle

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Ralf Küsters
Betreuer/in:	Dr. Kay Schweiger, Daniel Rausch, M.Sc.
Beginn am:	2. Mai 2019
Beendet am:	4. November 2019

Kurzfassung

Der 5G-Standard soll in den nächsten Jahren für deutlich schnelleres und sichereres mobiles Internet sorgen. Ein zentraler Bestandteil dieses Standards ist das 5G-AKA Protokoll. Es soll den sicheren Austausch eines kryptographischen Schlüssels und die Authentifizierung eines Benutzers ermöglichen und spielt somit eine wichtige Rolle bei der Sicherheit des 5G-Standards. Jedoch wurde bereits im Februar 2018 eine mögliche Sicherheitslücke in dem 5G-AKA Protokoll aufgedeckt, die dessen Sicherheit beeinträchtigen könnte. In dieser Arbeit wird daher ein Überblick über das 5G-AKA Protokoll und mögliche Auswirkungen dieser Sicherheitslücke gegeben.

Es wird das 5G-AKA Protokoll beschrieben und mit dem Vorgängerprotokoll des 4G-Standards verglichen. Des Weiteren wird die Sicherheitslücke und deren mögliche Auswirkungen beschrieben. Zuletzt wird die Sicherheitslücke implementiert um deren Auswirkungen in der Praxis zu testen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

5G-TMSI 5G Temporary Mobile Subscription Identifier. 20

5G SE AV 5G Serving Environment Authentication Vector. 18, 20

5G HE AV 5G Home Environment Authentication Vector. 18, 20

5G-GUTI 5G Globally Unique Temporary UE Identifier. 18, 20

ABBA Anti-Bidding down Between Architecture. 19, 24

AK Anonymity Key. 25

AMF Authentication Management Field. 18, 25

ARPF Authentication credential Repository and Processing Function. 16

ASME Access Security Management Entity. 35

AUSF Authentication Server Function. 15, 16

AUTH Authenticaton Token. 34

AUTN Authentication Token. 19, 20

AUTS Synchronization failure Parameter. 25

AV Authentication Vector. 34

CK Cipher Key. 25

eNodeB Evolved Node-B. 33

EPS Evolved Packet System. 25

f1, f2, f3, f4, f5 Message authentication and Key generating functions. 26

GUAMI Globally Unique AMF Identifier. 20

GUTI Globally Unique Temporary UE Identifier. 34

HRES* Hash Response*. 7, 21

HSS Home Subscriber Server. 33

HXRES* Hash Expected Response*. 7, 21

IK Integrity Key. 26

IMSI International Mobile Subscriber Identity. 26

K Authentication Key. 16, 21

K_{ASME} Key for ASME. 35

K_{AUSF} Key for AUSF. 18, 27

K_{SEAF} Key for SEAF. 17, 21

KDF Key Derivation Function. 27

MAC Message Authentication Code. 7, 27

MCC Mobile Country Code. 28

ME Mobile Equipment. 16

MME Mobility Management Entity. 33

MNC Mobile Network Code. 28

MSIN Mobile Subscriber Identification Number. 26

ngKSI Key Set Identifier in 5G. 19, 28

RAND Random challenge. 19, 22

RES Response. 28

RES* Response*. 7, 22

RRC Radio Resource Control. 34

SEAF Security Anchor Function. 15, 16

SIDF Subscription Identifier De-concealing Function. 16

SN-Id Serving Network Id. 22

SN-Name Serving Network Name. 18, 22

SQN Sequence Number. 28

SUCI Subscription Concealed Identifier. 23

SUPI Subscription Permanent Identifier. 24

SUPI Type Subscription Permanent Identifier Type. 29

UDM Unified Data Management. 15, 16

UE User Equipment. 15, 16

USIM Universal Subscriber Identity Module. 16

XMAC Expected Message Authentication Code. 7, 29

XRES Expected Response. 29

XRES* Expected Response*. 7, 24

1 Einleitung

Die Mobilfunknetze haben sich in den letzten Jahren stark weiterentwickelt. Vor allem die 5G Technologie soll deutliche Verbesserungen mit sich bringen. Dabei soll neben deutlich schnellerem Internet auch, im Gegensatz zu vorherigen Generationen, die Sicherheit spürbar verbessert werden. Um dies zu gewährleisten wurden neue Standards entwickelt. Bei diesen Standards handelt es sich um neue Protokolle, wie das 5G-AKA-Protokoll zum sicheren Austausch eines gemeinsamen kryptographischen Schlüssels und viele weitere Protokolle.

Was die Sicherheit angeht ist vor allem der sichere Austausch eines gemeinsamen kryptographischen Schlüssels von großer Bedeutung. In den früheren Generationen der Mobilfunktechnologien (2G, 3G und 4G) wurde die Sicherheit der Protokolle bereits weitreichend untersucht und auch für die Protokolle des 5G Standards wurden bereits einige Untersuchungen angestellt [Cab19].

1.1 4G-Standard

Der 4G-Standard ist die vierte Generation der Breitband-Mobilfunknetz-Technologie. Sein Vorgänger ist der 3G-Standard [KQAW09]. Der 4G-Standard verspricht, gegenüber dem 3G-Standard, eine deutlich schnellere Übertragungsrate von bis zu 100MBit/s und in manchen Situationen auch von bis zu 1GBit/s [ITU08]. Häufig wird 4G mit LTE (Long-Term Evolution) gleichgesetzt. Dies ist jedoch nicht ganz korrekt, da die Übertragungsrate des LTE-Standards die vom 4G-Standard geforderte Übertragungsrate leicht unterschreitet. Durch diese nur knappe Unterschreitung wird LTE von manchen Forschern auch als 3.9G bezeichnet [Kum11].

Mit LTE-A (LTE-Advanced) wurde eine Verbesserung des LTE-Standards entwickelt, der offiziell die Vorraussetzungen des 4G-Standards erfüllt [WKA+12]. Diese Verbesserung wurde von dem 3GPP (3rd Generation Partnership Project) eingebracht, das auch zu der Entwicklung des 5G-Standards maßgeblich beigetragen hat.

1.2 5G-Standard

Der 5G-Standard ist die fünfte Generation der Breitband-Mobilfunknetz-Technologie. Er ist der Nachfolger des 4G-Standards und der bisher neueste Standard auf dem Markt. Er wurde von dem 3GPP definiert.

Das 3GPP hat auch den 5G NR (5G New Radio) Standard entwickelt, der als RAT (Radio Access Technology) für mobile Netze gedacht ist [PDFF17]. Der 5G NR wird in zwei Modi der Öffentlichkeit zugänglich gemacht. Der erste Modus ist der NSA mode (Non-Standalone Mode), der von dem bisherigen LTE Netzwerk abhängt [Sut18]. Dieser Modus wird dann im Laufe der Zeit vom SA mode (Standalone Mode) abgelöst, der unabhängig von bisherigen Netzwerken funktioniert [Bro17].

1.3 Ziel der Arbeit

Ziel dieser Arbeit ist es, einen Überblick über das 5G-AKA-Protokoll zu erlangen. Dafür wird das Protokoll beschrieben und die Unterschiede bzw. Ähnlichkeiten zu dessen Vorgänger, dem 4G EPS-AKA-Protokoll, aufgezeigt. Des Weiteren wird auf die Sicherheit des Protokolls eingegangen, indem eine Sicherheitslücke dieses Protokolls beschrieben und implementiert wird [DC18].

1.4 Aufbau der Arbeit

In Kapitel 2 wird das 5G-AKA-Protokoll beschrieben. Dabei wird genauer auf dessen Funktionsweise eingegangen, sowie das Ziel des Protokolls erläutert. In Kapitel 3 wird das 5G-AKA-Protokoll mit dem 4G EPS-AKA-Protokoll des 4G Standards verglichen. Dafür wird das 4G EPS-AKA-Protokoll kurz beschrieben und dessen Ziele erläutert. Des Weiteren werden die Unterschiede der beiden Protokolle hervorgehoben und die Verbesserungen des neuen Protokolls gegenüber dem Alten aufgezeigt. Kapitel 4 gibt einen Überblick über eine Sicherheitslücke des 5G-AKA-Protokolls. Dafür wird die Sicherheitslücke beschrieben und deren Auswirkungen aufgezeigt. In Kapitel 5 wird die Implementierung der beschriebenen Sicherheitslücke dokumentiert und in Kapitel 6 ist eine Zusammenfassung dieser Arbeit und ein Ausblick zu finden.

2 Beschreibung des 5G-AKA Protokolls

Das 5G-AKA-Protokoll ist eines von mehreren Protokollen des 5G-Standards. Vor Allem das EAP-AKA-Protokoll und das 5G-AKA-Protokoll sind für den sicheren Austausch eines kryptographischen Schlüssels zuständig. Bei beiden Protokollen ist der erste Teil identisch, bis bei dem Provider eines der Protokolle ausgewählt wird [3GP19c, S. 40]. Bei beiden Protokollen ist auch der Authentifizierungsvorgang sehr ähnlich, wobei das 5G-AKA-Protokoll gegenüber dem EAP-AKA-Protokoll dahingehen erweitert wurde, dass es dem Provider, *Home Network* genannt, eine erfolgreiche Authentifizierung nachweist [3GP19c, S. 43].

Das 5G-AKA-Protokoll wird größtenteils in der Spezifikation TS 33.501 des 3GPP beschrieben. Dabei wird immer die zum Zeitpunkt dieser Arbeit neueste Version V15.34.1 (15.4.0) herangezogen [3GP19c].

2.1 Ziele

Ziel des 5G-AKA Protokolls ist es den Benutzer und das Netzwerk gegenseitig zu authentifizieren (*Mutual Authentication*) und Schlüsselmaterial bereitzustellen, die von nachfolgenden Protokollen verwendet werden können [3GP19c, S. 37]. Bei einer erfolgreichen Authentifizierung wissen alle Teilnehmer des Protokolls über den Erfolg Bescheid und es wurde ein gemeinsamer *Anchor Key* ausgetauscht. Aus diesem *Anchor Key* können weitere Schlüssel für mehrere Sicherheitskontexte abgeleitet werden. Ein Überblick über die ableitbaren Schlüssel ist in Abbildung 6.2.1-1 und 6.2.2-1 des 3GPP TS 33.501 V15.34.1 (15.4.0) zu finden [3GP19c, S. 49, 52].

2.2 Das Protokoll

2.2.1 Entitäten

In dem Protokoll werden vier grundlegende Entitäten beschrieben. Diese sind das User Equipment (UE), die Security Anchor Function (SEAF), die Authentication Server Function (AUSF) und das Unified Data Management (UDM).

User Equipment (UE)

Das UE kann sich zum Beispiel auf dem Smartphone des Nutzers oder auf einem 5G-USB Dongle befinden. Es lässt sich in das Mobile Equipment (ME) und das Universal Subscriber Identity Module (USIM) unterteilen. In dem USIM werden für die Authentifizierung benötigte Schlüssel und Benutzerkennungen gespeichert, die für die eindeutige Identifizierung und Authentifizierung des UE benötigt werden [3GP19c, S. 24]. In dem ME werden die Schlüssel, die für die Kommunikation nach dem 5G-AKA-Protokoll benötigt werden, berechnet.

Security Anchor Function (SEAF)

Die SEAF ist Teil des mit dem UE kommunizierenden Netzwerks, auch *Serving Network* genannt. Sie kommuniziert mit dem UE und mit dem Provider des Benutzers. Nach erfolgreicher Authentifizierung haben das UE und die SEAF beide einen gemeinsamen Schlüssel [3GP19c, S. 37].

Authentication Server Function (AUSF)

Die AUSF ist Teil des Providers, bei dem sich der Benutzer die hier verwendete SIM-Karte gekauft hat, auch *Home Network* genannt. Sie kommuniziert mit der SEAF und validiert die Antwort des UE für die SEAF.

Unified Data Management (UDM)

Das UDM ist, wie auch die AUSF Teil des *Home Network*. Grundsätzlich funktioniert es jedoch nicht ohne die Authentication credential Repository and Processing Function (ARPF) und die Subscription Identifier De-concealing Function (SIDF).

Das UDM und die ARPF generieren Authentifizierungsvektoren aus den Schlüsseln, die sie mit dem USIM teilen. Die SIDF berechnet die Benutzerkennung (SUPI) für das UDM und die ARPF aus einer verschlüsselten Benutzerkennung (SUCI), die sie von dem AUSF erhält.

2.2.2 Vorbedingungen

Damit das 5G-AKA Protokoll erfolgreich durchgeführt werden kann müssen folgende Vorbedingungen erfüllt sein [3GP19c]:

- Das USIM und das UDM/ARPF verfügen über den gemeinsamen Langzeitschlüssel Authentication Key (K).
- Die ARPF verfügt über den privaten Schlüssel und das USIM verfügt über den öffentlichen Schlüssel eines asynchronen Schlüsselpaars.

2.2.3 Authentifikationsprozedur

Die Abbildung 2.1 zeigt eine erfolgreiche Authentifizierung. Nach dem Ende der Prozedur haben sich alle Entitäten auf den *Anchor Key*, Key for SEAF (K_{SEAF}), geeinigt. Dieser wird als Grundlage für die weitere Kommunikation verwendet.

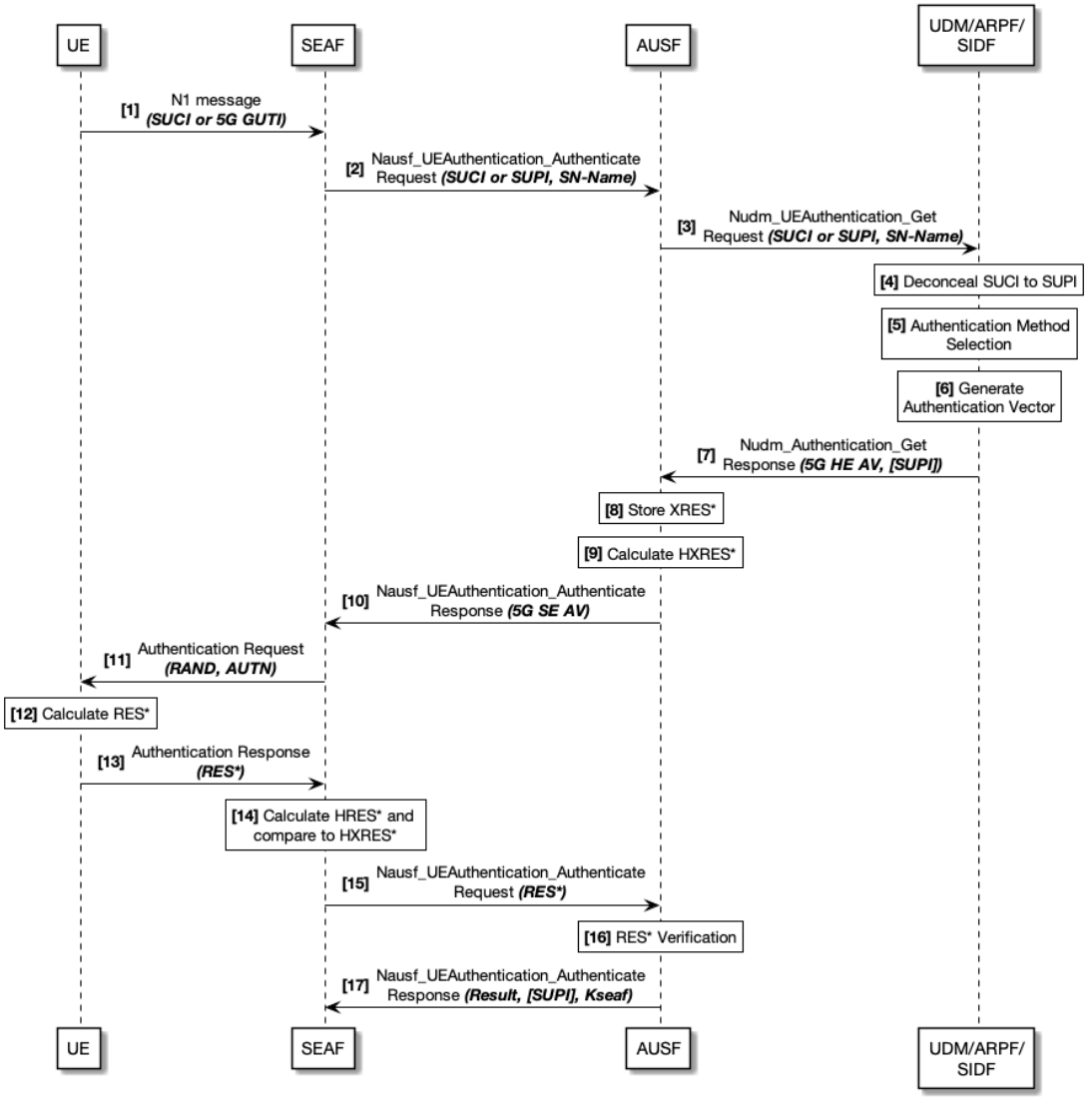

Abbildung 2.1: Das 5G-AKA Protokoll

Die einzelnen Nachrichten lassen sich folgendermaßen beschreiben [3GP19c, S. 40, 44]:

1. Das UE sendet in Schritt 1 die *N1 message* an die SEAF und beginnt damit die Authentifikationsprozedur. Diese Nachricht beinhaltet entweder den SUCI oder den 5G Globally Unique Temporary UE Identifier (5G-GUTI). Sie werden später von dem UDM/ARPF zur Authentifizierung des UE verwendet.
2. Nach dem Empfang der *N1 message* sendet die SEAF in Schritt 2 den *Nausf_UEAuthentication_Authenticate Request* an die AUSF. Falls in der *N1 message* der 5G-GUTI enthalten ist und es sich um eine Re-Authentifizierung handelt, dann soll der SUPI an die AUSF gesendet werden, ansonsten wird der SUCI weitergeleitet. Zusätzlich wird noch der Serving Network Name (SN-Name) mitgesendet. Dies ist nötig, da das UDM den SN-Name für die Generierung des 5G Home Environment Authentication Vector (5G HE AV) benötigt.
3. Nach dem Empfang des *Nausf_UEAuthentication_Authenticate Request*s prüft die AUSF, ob die SEAF berechtigt ist, den gesendeten SN-Name zu verwenden. Falls dieser Test erfolgreich war, sendet die AUSF den *Nudm_UEAuthentication_Get Request* an das UDM. Er enthält den SUCI oder SUPI und den SN-Name, den die AUSF aus *Schritt 2* erhalten hat. Falls der Test nicht erfolgreich war, antwortet die AUSF mit der *Nausf_UEAuthentication_Authenticate Response*, die den Text *serving network not authorized* enthält.
4. Falls die AUSF in *Schritt 3* den SUCI an das UDM gesendet hat, wird in diesem Schritt aus dem SUCI der SUPI abgeleitet. Hierfür wird die SIDF verwendet.
5. In diesem Schritt wird die Authentifikationsmethode ausgewählt. Das UDM kann zwischen dem EAP-AKA' und dem 5G-AKA Protokoll wählen. Da in dieser Ausarbeitung nur auf das 5G-AKA-Protokoll eingegangen wird, wird angenommen, dass hier immer das 5G-AKA-Protokoll als Authentifikationsmethode gewählt wird.
6. Nun wird der 5G HE AV generiert. Hierfür wird der SN-Name und der SUPI benötigt, den das UDM in *Schritt 3* erhalten oder in *Schritt 4* erzeugt hat. Für die Generierung soll das *separation bit* des Authentication Management Field (AMF) auf "1" gesetzt werden.
7. Jetzt sendet das UDM die *Nudm_Authentication_Get Response* an die AUSF. Sie beinhaltet den 5G HE AV, der in *Schritt 6* generiert wurde, und den Hinweis, dass der 5G HE AV für das 5G-AKA-Protokoll bestimmt ist. Falls das UDM in *Schritt 3* den SUCI erhalten und in *Schritt 4* den SUPI erzeugt hat, dann wird der SUPI in der *Nudm_Authentication_Get Response* mitgeschickt.
8. In Schritt 8 wird die XRES*, die in dem 5G HE AV aus *Schritt 7* enthalten ist, und der SUCI oder SUPI zwischengespeichert, da sie in *Schritt 16* für die Verifikation von RES* benötigt werden. Des Weiteren wird auch der Key for AUSF (K_{AUSF}), welcher auch in dem 5G HE AV enthalten ist, zwischengespeichert.
9. Als nächstes wird aus der XRES* die HXRES* gerechnet. Des Weiteren wird der 5G Serving Environment Authentication Vector (5G SE AV) für *Schritt 10* aus dem 5G HE AV berechnet. Dafür wird die XRES* durch die HXRES* ersetzt und der K_{AUSF} entfernt.
10. Nachdem der 5G SE AV in *Schritt 9* erzeugt wurde wird er in Schritt 10 mit der *Nausf_UEAuthentication_Authenticate Response* an die SEAF gesendet. Erhält die SEAF diese Nachricht nicht, so soll mit *Schritt 2'* von Abbildung 2.3 fortgefahren werden.

11. Nach dem Erhalten der *Nausf_UEAuthentication_Authenticate Response* wird die SEAF in Schritt 11 den *Authentication Request* an das UE schicken. Er beinhaltet die Random challenge (RAND) und das Authentication Token (AUTN) den die SEAF in *Schritt 10* erhalten hat. Zusätzlich wird noch der Key Set Identifier in 5G (ngKSI) und die Anti-Bidding down Between Architecture (ABBA), die nach einer erfolgreichen Authentifizierung benötigt werden, mitgesendet. Des Weiteren wird die HXRES*, die in dem 5G SE AV enthalten ist, bei der SEAF zwischengespeichert, da sie noch für *Schritt 14* benötigt wird.

12. Nachdem das UE den *Authentication Request* von der SEAF erhalten hat, wird in Schritt 12 überprüft ob der AUTN angenommen werden kann. Ist dies der Fall, dann wird aus der RAND und dem AUTN die RES* für die *Authentication Response* berechnet. Hierfür wird das *separation bit* des AMF auf ”1” gesetzt. Es wird auch der XMAC berechnet und mit dem MAC, der in dem AUTN enthalten ist, verglichen. Stimmen XMAC und MAC überein, so wird die Authentifizierung fortgesetzt. Stimmen sie nicht überein oder kann der AUTN nicht angenommen werden, so wird die Authentifizierung als fehlgeschlagen angesehen und es wird die Prozedur aus Abbildung 2.2 ausgeführt.

13. In Schritt 13 wird die RES*, die in *Schritt 12* berechnet wurde, in der *Authentication Response* vom UE an die SEAF gesendet.

14. Nachdem die SEAF die *Authentication Response* vom UE erhalten hat, berechnet sie aus der RES* die HRES* und vergleicht sie mit der HXRES*, die sie in *Schritt 11* zwischengespeichert hat. Stimmen sie überein so wird die Authentifizierung von der SEAF als erfolgreich angesehen und es geht weiter mit *Schritt 15*. Stimmen sie nicht überein, so wird die Authentifizierung von der SEAF als fehlgeschlagen angesehen und es wird die Prozedur aus dem Abschnitt *HRES* failure in SEAF* in Abbildung 2.3 ausgeführt. Wird keine Antwort vom UE erhalten, so wird die Authentifizierung von der SEAF als fehlgeschlagen angesehen und die AUSF wird über den Fehlschlag informiert. Wann dieser Zeitpunkt erreicht ist, ist nicht spezifiziert.

15. In Schritt 15 sendet die SEAF den *Nausf_UEAuthentication_Authenticate Request* an die AUSF. Er beinhaltet die RES* den die SEAF in der *Authentication Response* vom UE erhalten hat.

16. Nachdem die AUSF den *Nausf_UEAuthentication_Authenticate Request* erhalten hat vergleicht sie die RES* mit der XRES*, die sie in *Schritt 8* zwischengespeichert hat. Stimmen sie überein so wird die Authentifizierung von der AUSF als erfolgreich angesehen und es geht weiter mit *Schritt 17*. Stimmen sie nicht überein, so wird die Authentifizierung von der AUSF als fehlgeschlagen angesehen und es wird die Prozedur aus dem Abschnitt *RES* failure in SEAF* in Abbildung 2.3 ausgeführt. Außerdem kann auch überprüft werden, ob der Authentication Vector abgelaufen ist. Dies ist aber nicht zwingend vorgeschrieben. Die AUSF informiert das UDM über das Ergebnis (*Result*) des Vergleichs.

17. In Schritt 17 sendet die AUSF die *Nausf_UEAuthentication_Authenticate Response* an die SEAF. Sie beinhaltet das Ergebnis des Vergleichs von RES* und XRES* aus *Schritt 16*. Stimmen RES* und XRES* überein, wird auch der *Anchor Key* K_{SEAF} mitgesendet. Wenn die AUSF in *Schritt 7* den SUPI erhalten hat, dann wird im Falle des erfolgreichen Vergleichs von RES* und XRES* der SUPI nun auch an die SEAF weitergeleitet.

2.2.4 Wichtige Definitionen

5G Globally Unique Temporary UE Identifier (5G-GUTI)

Der 5G-GUTI ist ein Bitfeld das vom AMF erstellt wird. Er kann, wie auch der SUCI, für die Initiierung der Authentifizierung verwendet werden. Er besteht aus zwei Teilen mit denen man das AMF und das UE eindeutig identifizieren kann [3GP19c, S. 35]. Diese Teile sind der Globally Unique AMF Identifier (GUAMI) und der 5G Temporary Mobile Subscription Identifier (5G-TMSI).

$$5G-GUTI = GUAMI||5G-TMSI$$

$$GUAMI = MCC||MNC||AMFIdentifier$$

$$AMFIdentifier = AMFRegionId||AMFSetId||AMFPointer$$

Der 5G-TMSI hat eine Länge von 32 Bits, die AMF Region Id hat eine Länge von 8 Bits, die AMF Set Id hat eine Länge von 10 Bits und der AMF Pointer hat eine Länge von 6 Bits [3GP19a, S. 36].

5G Home Environment Authentication Vector (5G HE AV)

Der 5G HE AV ist eine Ansammlung von Zahlen und Bitfeldern, die für die Authentifizierung benötigt werden. Er wird von dem UDM erstellt und an die AUSF geschickt.

Er beinhaltet RAND, AUTN, XRES* und K_{AUSF} [3GP19c, S. 44].

5G Serving Environment Authentication Vector (5G SE AV)

Der 5G SE AV ist ebenfalls eine Ansammlung von Zahlen und Bitfeldern, die für die Authentifizierung benötigt werden. Er wird von der AUSF erstellt und an die SEAF geschickt.

Er beinhaltet RAND, AUTN und HXRES* [3GP19c, S. 44].

Authentication Token (AUTN)

Der AUTN ist ein Bitfeld und besteht aus den vier Komponenten SQN, AK, AMF und MAC. Er ist wie folgt aufgebaut [3GP18a, S. 25]:

$$AUTN = (SQN \oplus AK)||AMF||MAC$$

- Das AMF wird benötigt, um MAC zu berechnen und spezifische Anpassungen vorzunehmen.
- Der MAC wird benötigt, um ihn mit dem XMAC zu vergleichen.
- Die SQN⊕AK soll die SQN übermitteln. Sie wird mit dem AK verschlüsselt, um die Identität und Position der Benutzers nicht preiszugeben. Ist AK = 0, dann kann davon ausgegangen werden, dass die SQN alleine weder Identität noch Position des Benutzers preisgibt.

Hash Response* (HRES*)

HRES* ist der Hash von RES* und wird von der SEAF benötigt, um ihn mit dem HXRES* zu vergleichen. Er wird von der SEAF mit Hilfe des SHA-256-Algorithmus' berechnet [3GP19c, S. 155]. Eingabe des SHA-256-Algorithmus' ist die Konkatenation von RAND und RES* (RAND || RES*).

HRES* setzt sich aus den 128 *least significant* Bits von der Ausgabe des SHA-256 zusammen.

Hash Expected Response* (HXRES*)

HXRES* ist der Hash von XRES* und wird von der SEAF benötigt, um ihn mit dem HRES* zu vergleichen. Er wird von der AUSF mit Hilfe des SHA-256-Algorithmus' berechnet [3GP19c, S. 155]. Eingabe des SHA-256-Algorithmus' ist die Konkatenation von RAND und XRES* (RAND || XRES*).

HXRES* setzt sich aus den 128 *least significant* Bits von der Ausgabe des SHA-256 zusammen.

Authentication Key (K)

Der Schlüssel K hat eine Länge von 128 oder 256 Bits und wird für die Generierung von AK, CK, IK RES bzw. XRES und MAC bzw. XMAC benötigt [3GP18a, S. 32]. Die tatsächliche Länge ist nicht spezifiziert und wird möglicherweise vom Provider festgelegt.

K soll auf dem UDM/ARPF und in einer manipulationssicheren Hardwarekomponente auf dem UE gespeichert werden [3GP19c, S. 29].

Key for SEAF (K_{SEAF})

Der K_{SEAF}, auch *Anchor Key* genannt, ist der Schlüssel auf den sich alle Entitäten bei erfolgreicher Authentifizierung einigen [3GP19c, S. 37]. Er wird von der AUSF und dem UE berechnet und lässt sich mit Hilfe der KDF berechnen. Dabei sind folgende Werte Eingabeparameter der KDF [3GP19c, S. 155]:

- KEY = K_{AUSF}
- Fc = 0x6C
- P0 = SN-Name

Die Ausgabe der KDF ist der K_{SEAF}.

Random challenge (RAND)

Der RAND ist eine 128 Bit lange zufällige Zahl [3GP18a, S. 32]. Sie wird für die Generierung von AK, CK, IK RES bzw. XRES und MAC bzw. XMAC benötigt. Der RAND soll eine unvorhersehbare Zahl sein, wie genau sie generiert wird ist jedoch nicht beschrieben [3GP18a, S. 25].

Sie wird vom UDM/ARPF erzeugt und an das UE weitergeleitet. Diese Entitäten generieren auch die oben genannten Schlüssel AK, CK, IK, usw.

Response* (RES*)

Die RES* ist eine Bitfolge und lässt sich, wie auch XRES*, mit Hilfe der KDF berechnen. Dabei sind folgende Werte Eingabeparameter der KDF [3GP19c, S. 155]:

- KEY = Konkatenation von CK und IK (CK || IK)
- Fc = 0x6B
- P0 = SN-Name
- P1 = RAND
- P2 = RES

RES* setzt sich aus den 128 *least significant* Bits von der Ausgabe der KDF zusammen. RES* wird für die Berechnung von HRES* benötigt. Es wird vom UE berechnet.

Serving Network Name (SN-Name)

Der SN-Name wird für die Berechnung von RES* bzw. XRES*, K_{AUSF} und K_{SEAF} benötigt. Er hat zwei Zwecke [3GP19c, S. 39]:

- Er bindet den K_{SEAF}, auch *Anchor Key* genannt, an die SEAF, auch *Serving Network* genannt, indem er die Serving Network Id (SN-Id) beinhaltet.
- Er stellt sicher, dass der *Anchor Key* nur für die Authentifizierung in 5G-Netzwerken verwendbar ist, indem er den Service Code ”5G” beinhaltet.

Der SN-Name ist wie folgt aufgebaut [3GP19c, S. 39]:

Er beginnt mit dem Service code ”5G”, gefolgt von dem Trennzeichen ”:” und der SN-Id. Die SN-Id ist für jedes *Serving Network* eindeutig.

$$SN - Name = '' 5G'' || '' : '' || SN - Id$$

$$SN - Id = MCC || MNC$$

Subscription Concealed Identifier (SUCI)

Der SUCI ist die verschlüsselte Version des SUPI und wird für die Identifizierung des UE benötigt.. Er lässt sich mit der SIDF in den SUPI umwandeln.

Er ist wie folgt aufgebaut [3GP19a, S. 27]:

SUCI = SUPI Type || Home Network Identifier || Routing Indicator || Protection Scheme Id || Home Network Public Key Id || Scheme Output

- Der Home Network Identifier ist von dem SUPI Type abhängig.
  - Wenn der SUPI Type den IMSI angibt, dann entspricht der Home Network Identifier dem MCC, gefolgt von dem MNC (Home Network Identifier = MCC || MNC).
  - Wenn der SUPI Type den Network Specific Identifier angibt, dann entspricht der Home Network Identifier einer ähnlichen Form des Network Access Identifiers, wie er beispielsweise auch beim SUPI verwendet wird. (Home Network Identifier = username@realm) Er ist genauer im Dokument IETF RFC 7542 spezifiziert [Int15, S. 10].
- Der Routing Indicator besteht aus einem bis zu vier Zeichen [3GP19a, S. 28]. Er erlaubt es der SEAF zusammen mit dem Home Network Identifier eine Verbindung mit der korrekten AUSF aufzubauen. Falls kein Routing Indicator konfiguriert ist soll er auf 0 gesetzt werden.
- Der Protection Scheme Identifier kann Werte von 0 bis 15 annehmen. Die genaue Definition der einzelnen Algorithmen ist im Annex C des 3GPP TS 33.501 V15.34.1 zu finden [3GP19c, S. 168].
  - Der Wert 0 bedeutet, dass das Null Scheme verwendet wird.
  - Der Wert 1 bedeutet, dass das Profile A verwendet wird.
  - Der Wert 2 bedeutet, dass das Profile B verwendet wird.
  - Die Werte 3 bis 11 sind für zukünftige Spezifikationen reserviert.
  - Die Werte 12 bis 15 sind für spezifische Algorithmen reserviert.
- Der Home Network Public Key Identifier kann Werte von 0 bis 255 annehmen. Er wird genutzt um den Public Key zu identifizieren, der für die Verschlüsselung des SUPI verwendet wird.
- Der Scheme Output ist die Ausgabe des Verschlüsselungs-Algorithmus. Die genaue Definition des Scheme Outputs ist im Annex C des 3GPP TS 33.501 V15.34.1 zu finden [3GP19c, S. 168].

Subscription Permanent Identifier (SUPI)

Der SUPI wird für die Identifizierung des UE benötigt. Er ist für jeden Nutzer global eindeutig und ist wie folgt aufgebaut [3GP19a, S. 27]:

SUPI = SUPI Type || Home Network Identifier

Der Identifier ist von dem SUPI Type abhängig und kann folgende Werte annehmen:

- Wenn der SUPI Type den IMSI angibt, dann entspricht der Home Network Identifier dem IMSI. (Home Network Identifier = IMSI)
- Wenn der SUPI Type den Network Specific Identifier angibt, dann entspricht der Home Network Identifier dem Network Access Identifier. (Home Network Identifier = Network Access Identifier) Der Network Access Identifier hat die Form $username@realm$. Er ist genauer im Dokument IETF RFC 7542 spezifiziert [Int15, S. 10].

Expected Response* (XRES*)

Die XRES* ist eine Bitfolge und lässt sich, wie auch RES*, mit Hilfe der KDF berechnen. Dabei sind folgende Werte Eingabeparameter der KDF [3GP19c, S. 155]:

- KEY = Konkatenation von CK und IK (CK || IK)
- Fc = 0x6B
- P0 = SN-Name
- P1 = RAND
- P2 = XRES

XRES* setzt sich aus den 128 *least significant* Bits von der Ausgabe der KDF zusammen. XRES* wird für die Berechnung von HXRES* benötigt. Es wird von der ARPF berechnet.

2.2.5 Weitere Begriffe

Anti-Bidding down Between Architecture (ABBA)

Der ABBA Parameter soll einen *Anti-bidding down*-Schutz für Sicherheitsfeatures gewährleisten, der verhindern soll, dass ältere Sicherheitsfeatures verwendet werden obwohl alle Parteien auch neuere Sicherheitsfeatures unterstützen [3GP19c, S. 22]. Außerdem soll damit angegeben werden welche Sicherheitsfeatures in diesem Netzwerk aktiv sind.

Der ABBA Parameter wird von der SEAF an das UE gesendet und hat eine variable Länge. Zurzeit kann der ABBA Parameter nur den Wert 0x0000 annehmen. Dieser Wert steht für die bisher definierten Sicherheitsfeatures [3GP19c, S. 156].

Anonymity Key (AK)

Der AK ist ein Schlüssel mit einer Länge von 48 Bits. Er wird zum Verschlüsseln der SQN verwendet [3GP18a, S. 32]. Lässt die SQN keine Rückschlüsse auf die Identität oder Position des Nutzers zu, so kann der AK auch auf 0 gesetzt werden [3GP18a, S. 25].

Der AK wird mit der *Key generating function* f5 berechnet und bekommt RAND und K als Eingabe [3GP18a, S. 26].

Authentication Management Field (AMF)

Das AMF ist ein 16 Bit langes Bitfeld [3GP18a, S. 32]. Es wird für die Berechnung von Tokens und anderen Bitfeldern benötigt. Die 16 Bits des AMF sind von 0 bis 15 durchnummeriert [3GP18a, S. 79]. 0 steht hier für das *most significant bit* und 15 steht für das *least significant bit*.

- Bit 0, auch *separation bit* genannt, wird für das Evolved Packet System (EPS) verwendet und wird bei der Verwendung des AMF meist explizit angegeben.

- Bits 1 bis 7 sind für eine zukünftige Spezifikation reserviert und sollen auf 0 gesetzt werden.

- Bits 8 bis 15 können für proprietäre Zwecke verwendet werden, sind aber nicht explizit festgelegt. Sie können verwendet werden um spezifische Anpassungen festzulegen. Zum Beispiel könnten mehrere Authentifikations-Algorithmen unterstützt werden, indem die Bits 8 bis 15 den Algorithmus festlegen oder es könnten die Verifikationsparameter der SQN durch die Bits bestimmt werden [3GP18a, S. 77].

Synchronization failure Parameter (AUTS)

Der AUTS Parameter wird für die Re-Synchronisations Prozedur aus *Schritt 3** von Abbildung 2.2 benötigt. Die SEAF erhält den AUTS Parameter von dem UE nach einer beim UE fehlgeschlagenen Authentifizierung.

$$AUTS = Conc(SQN)||MACS$$

$$Conc(SQN) = SQN \oplus f5*(RAND)$$

$$MACS = f1*(SQN||RAND||AMF)$$

Die Funktionen f1* und f5* sind, wie auch die *Message authentication and Key generating functions* vom K abhängig, dürfen aber keine Schlüsse über die Funktionen f1, f2, f3, f4, f5 zulassen.

Cipher Key (CK)

Der CK ist ein 128 Bit langer Schlüssel [3GP18a, S. 32]. Er wird, zusammen mit dem IK, benötigt um den K_{AUSF} zu berechnen.

Der CK wird von der *Key generating function* f3 berechnet. Diese Funktion bekommt RAND und K als Eingabe [3GP18a, S. 26].

Message authentication and Key generating functions (f1, f2, f3, f4, f5)

Mit diesen Funktionen werden AK, CK, IK RES bzw. XRES und MAC bzw. XMAC berechnet [3GP18a, S. 26].

- Bei f1 handelt es sich um eine *Message authentication function*. Sie berechnet den MAC bzw. XMAC und bekommt K und die Konkatenation von SQN, RAND und AMF (SQN || RAND || AMF) als Eingabe.

- Bei f2 handelt es sich um eine, möglicherweise verkürzte, *Message authentication function*. Sie berechnet die RES bzw. XRES und bekommt K und RAND als Eingabe.
- Bei f3, f4 und f5 handelt es sich um *Key generating function*s. f3 berechnet CK, f4 berechnet IK und f5 berechnet AK. Alle drei Funktionen bekommen K und RAND als Eingabe. Falls SQN keine Rückschlüsse auf die Identität oder Position des Benutzers zulässt, wird f5 = 0 gesetzt.

Integrity Key (IK)

Der IK ist ein 128 Bit langer Schlüssel [3GP18a, S. 32]. Er wird zusammen mit dem CK benötigt, um K_{AUSF} zu berechnen.

Er wird von der *Key generating function* f4 berechnet. Diese Funktion bekommt RAND und K als Eingabe [3GP18a, S. 26].

International Mobile Subscriber Identity (IMSI)

Die IMSI ist eine bis zu 15 Zahlen lange Zahlenfolge und wird unter anderem für die Identifizierung des UE benötigt. Die IMSI darf nur aus den Dezimalzahlen 0 - 9 bestehen. Sie wird aus dem MCC, dem MNC und der Mobile Subscriber Identification Number (MSIN) aufgebaut [3GP19a, S. 26, 29].

$$IMSI = MCC||MNC||MSIN$$

Die MSIN ist bis zu 9 Zahlen lang und identifiziert den Benutzer eindeutig.

Key for AUSF (K_{AUSF})

Der K_{AUSF} wird für die Berechnung des K_{SEAF} benötigt. Er lässt sich mit Hilfe der KDF berechnen. Dabei sind folgende Werte Eingabeparameter der KDF [3GP19c, S. 154]:

- KEY = Konkatenation von CK und IK (CK || IK)
- Fc = 0x6A
- P0 = SN-Name
- P1 = Konkatenation von SQN und AK (SQN || AK).

Die Ausgabe der KDF ist der K_{AUSF}.

K_{AUSF} wird von der ARPF und dem UE berechnet und wird für die Berechnung von K_{SEAF} benötigt.

Key Derivation Function (KDF)

Die KDF wird für die Berechnung von K_{AUSF}, K_{SEAF} und RES* bzw. XRES* benötigt.

Die Eingabeparameter der KDF sind KEY, Fc, P0, . . . , Pn und optional auch L0, . . . , Ln.

Um den Schlüssel zu berechnen muss zuerst der Wert S berechnet werden. Er lässt sich folgendermaßen zusammensetzen [3GP18b, S. 46]:

$$S = Fc||P0||L0||P1||L1|| \ldots ||Pn||Ln$$

Die Länge von allen P1, . . . , Pn ist durch 8 teilbar. L1, . . . , Ln sind genau 16 Bit groß und geben die Längen der Parameter P1, . . . , Pn an. Das heißt, dass Li die Anzahl der Bytes, die mindestens benötigt werden um Pi zu speichern, binär in 16 Bits kodiert.
Zum Beispiel: Ist P0 16 Bit und P1 24 Bit lang, dann hat L0 den Wert 0x02 und L1 den Wert 0x03.

Die Ausgabe der KDF ist das Ergebnis folgender Rechnung:
Ausgabe = HMAC-SHA-256(KEY, S)

Message Authentication Code (MAC)

Der MAC hat eine Länge von 64 Bits und wird vom UE benötigt um ihn mit dem XMAC zu vergleichen. Er wird, wie auch der XMAC, mit der *Message authentication function* f1 berechnet [3GP18a, S. 32]. Die Funktion f1 ist von K abhängig und bekommt als Eingabe die Konkatenation von SQN, RAND und AMF (SQN || RAND || AMF) [3GP18a, S. 26]. Der MAC wird von der UDM/ARPF berechnet.

Mobile Country Code (MCC)

Der MCC ist drei Zeichen lang und ist Teil der IMSI und des SN-Name [3GP19a, S. 26]. Er identifiziert das Land in dem sich der Provider befindet.

Mobile Network Code (MNC)

Der MNC ist zwei oder drei Zeichen lang, wie auch der MCC, Teil der IMSI und des SN-Name [3GP19a, S. 26]. Die genaue Länge ist vom MCC abhängig und somit von Land zu Land unterschiedlich. Der MNC und der MCC zusammen identifizieren den Provider.

Key Set Identifier in 5G (ngKSI)

Der ngKSI identifiziert den K_{AMF} eindeutig [3GP19c, S. 54]. Er wird von der SEAF an das UE geschickt und wird nach einer erfolgreichen Authentifizierung benötigt um den K_{AMF} zu identifizieren.

Response (RES)

Die RES ist eine Bitfolge mit einer variablen Länge von bis zu 416 Bytes. Sie wird von dem UE mit der *Message authentication function* f2 berechnet. Die Funktion bekommt K und RAND als Eingabe [3GP18a, S. 26, 32].

Die RES wird von der AUSF mit der XRES verglichen. Stimmen sie nicht überein so bricht die AUSF die Authentifizierung ab [3GP19c, S. 45].

Sequence Number (SQN)

SQN ist eine Zahl mit einer Länge von 48 Bits. Aus ihr wird der MAC bwz. XMAC erzeugt.

Für die Generierung der SQN muss das UDM/ARPF folgende Bedingungen erfüllen [3GP18a, S. 25, 32]:

1. Der Erzeugungsmechanismus der SQN soll eine Re-Synchronisations Prozedur beinhalten. Die Re-Synchronisations Prozedur ist eine Prozedur die ausgeführt werden kann, wenn der AUTN vom UE in *Schritt 1** von Abbildung 2.2 nicht angenommen werden kann (*Synchronization failure*).
2. Der Erzeugungsmechanismus muss gegen den Überlauf der SQN (*Wrap Around Protection*) geschützt sein, da sonst eine bereits verwendete SQN möglicherweise erneut erzeugt und verwendet werden könnte.

Falls die SQN Rückschlüsse auf die Identität oder Position des Nutzers zulässt, soll der AK verwendet werden um die SQN zu verbergen.

Subscription Permanent Identifier Type (SUPI Type)

Der SUPI Type wird zum Erzeugen des SUPI benötigt und kann Werte von 0 bis 7 annehmen [3GP19a, S. 27].

- Der Wert 0 bedeutet, dass der Home Network Identifier den IMSI angibt.
- Der Wert 1 bedeutet, dass der Home Network Identifier den Network Specific Identifier angibt.
- Die Werte 2 bis 7 sind für zukünftige Spezifikation reserviert.

Expected Message Authentication Code (XMAC)

Der XMAC hat eine Länge von 64 Bits und wird vom UE benötigt um ihn mit dem MAC zu vergleichen. Er wird, wie auch der MAC, mit der Funktion f1 berechnet. Die *Message authentication function* f1 bekommt als Eingabe K und die Konkatenation von SQN, RAND und AMF (SQN || RAND || AMF) [3GP18a, S. 27, 32].

Der XMAC wird von dem UE berechnet und mit dem MAC verglichen. Den MAC hat das UE bereits von der SEAF erhalten. Stimmen MAC und XMAC nicht überein, so ist die Authentifizierung fehlgeschlagen [3GP18a, S. 27].

Expected Response (XRES)

Die XRES ist eine Bitfolge mit einer variablen Länge von bis zu 416 Bytes. Sie wird, wie auch die RES, mit der *Message authentication function* f2 berechnet. Diese Funktion bekommt K und RAND als Eingabe [3GP18a, S. 26, 32].

Die AUSF vergleicht RES und XRES miteinander. Stimmen sie nicht überein, so bricht die AUSF die Authentifizierung ab [3GP19c, S. 45].

2.2.6 Fehlgeschlagene Authentifikation

Fehlschlag beim Vergleich von MAC und XMAC [3GP19c, S. 46]

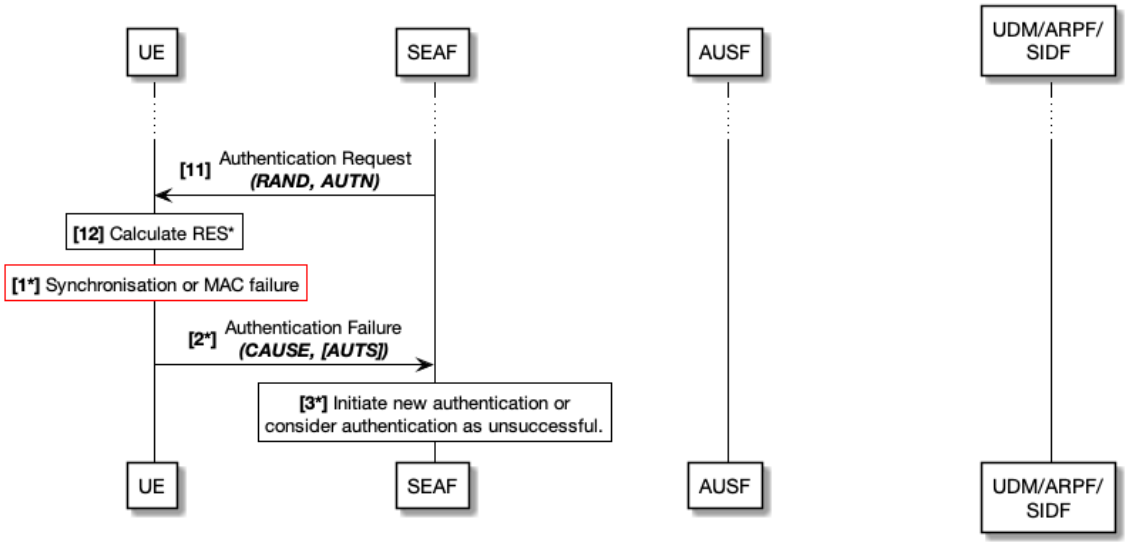

Abbildung 2.2: Fehlschlag beim Vergleich von MAC und XMAC

1*. In Schritt 1* wird entweder festgestellt, dass der berechnete XMAC nicht mit dem MAC, der in Schritt 11 empfangen wurde übereinstimmt (*MAC failure*) oder dass der AUTN nicht angenommen werden kann (*Synchronisation failure*).

2*. In Schritt 2* sendet das UE die *Authentication Failure* an die SEAF. Sie beinhaltet den CAUSE Wert, welcher den Grund für das Fehlschlagen beschreibt. Handelt es sich um eine *Synchronisation failure* so wird auch der AUTS Parameter mitgesendet.

3*. In Schritt 3* wird entschieden ob die Authentifikation gescheitert ist oder ob eine neue Authentifikation initiiert werden soll. Eine neue Authentifikation kann nur initiiert werden wenn es sich bei dem Fehlschlag um eine *Synchronisation failure* handelt. Die genaue Beschreibung der Re-Authentifizierungsprozedur ist im Dokument TS 24.501 zu finden [3GP19b].

Fehlschlag beim Vergleich von RES* bzw. HRES* mit XRES* bzw. HXRES* [3GP19c, S. 45]

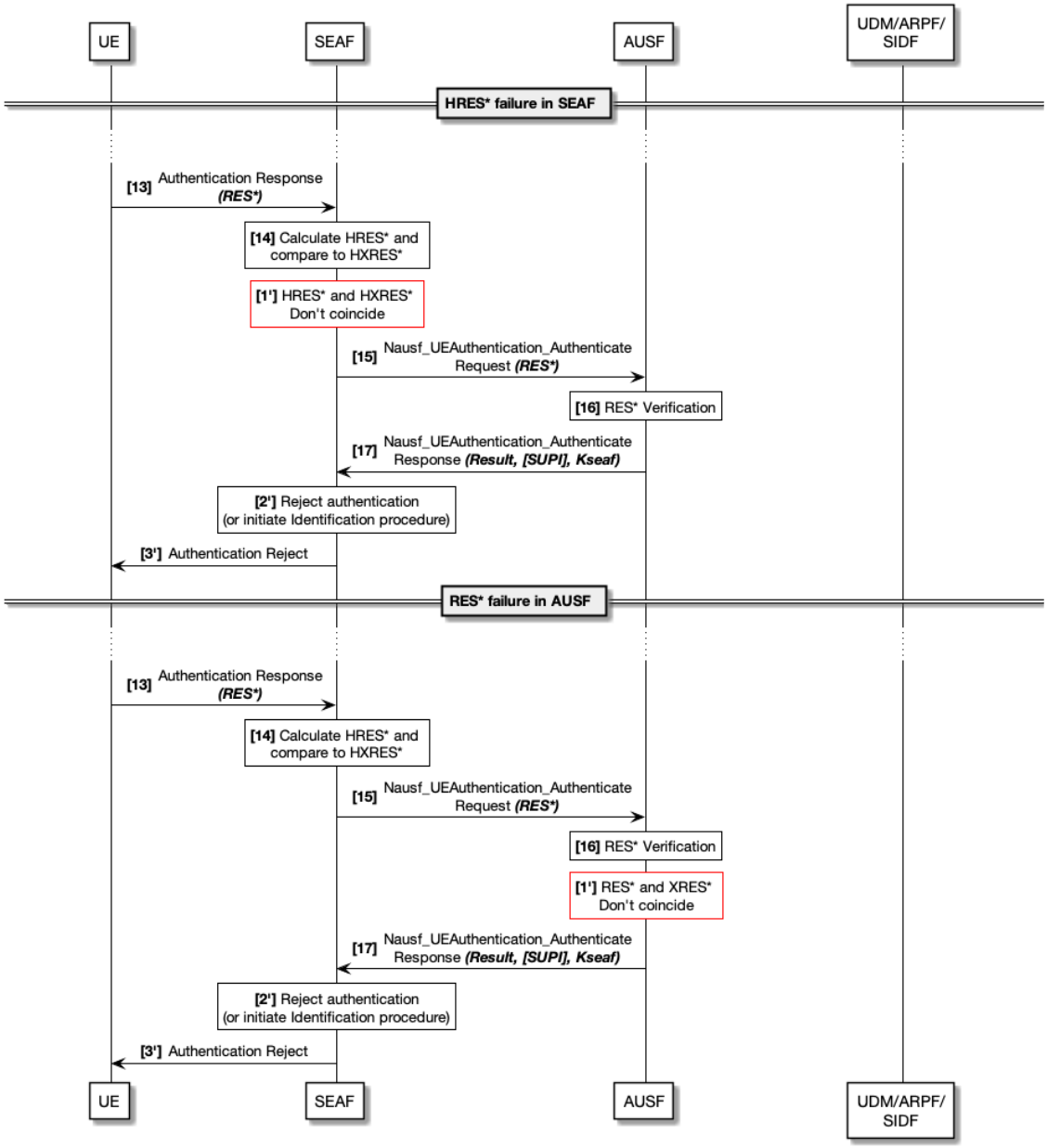

Abbildung 2.3: Fehlschlag beim Vergleich von RES* bzw. HRES* mit XRES* bzw. HXRES*

1'. Im Abschnitt *HRES* failure in SEAF* wird bei Schritt 1' festgestellt, dass HRES* und HXRES*, die in *Schritt 14* verglichen wurden, nicht übereinstimmen. Es wird trotzdem die Nachricht aus *Schritt 15* gesendet und auf die Antwort der AUSF gewartet.
Im Abschnitt *RES* failure in AUSF* wird bei Schritt 1' festgestellt, dass RES* und XRES*, die in *Schritt 16* verglichen wurden, nicht übereinstimmen. Dies wird der SEAF in *Schritt 17* mit dem *Result Parameter* mitgeteilt.

2'. Im Schritt 2' wird die Authentifizierung als nicht erfolgreich angesehen und es wird entweder eine neue Authentifizierung initiiert oder die Authentifizierung wird abgebrochen. Dies ist der Fall wenn der Vergleich aus dem *Schritt 1'* im Abschnitt *RES* failure in AUSF* oder im Abschnitt *HRES* failure in SEAF* fehlschlägt. Schlägt er in *Schritt 1'* aus Abschnitt *HRES* failure in SEAF* fehl, so wird unabhängig von dem *Result Parameter* aus *Schritt 17* die Authentifizierung als nicht erfolgreich angesehen. Ebenso wird die Authentifizierung als nicht erfolgreich angesehen wenn die Nachricht aus *Schritt 10* nicht von der SEAF erhalten wird. Die Authentifizierung wird abgebrochen wenn die SEAF in *Schritt 1* den SUCI erhalten hat. Falls die SEAF in *Schritt 1* den 5G-GUTI erhalten hat, so soll die *Identification procedure* gestartet werden, durch die die SEAF den SUCI vom UE erhalten soll.

3'. Falls in *Schritt 1* die SEAF den SUCI erhalten hat soll in Schritt 3' die *Authentication Reject* Nachricht an das UE gesendet werden.

3 Vergleich von 4G EPS-AKA und 5G-AKA

Das 4G EPS-AKA Protokoll ist Teil des 4G Standards, dem Vorgänger des 5G Standards, zu dem das 5G-AKA Protokoll gehört [Cab19]. Beide Protokolle haben Gemeinsamkeiten und Unterschiede, die in diesem Kapitel genauer erläutert werden.

3.1 Beschreibung des 4G EPS-AKA Protokolls

Das 4G EPS-AKA Protokoll ist ein *Authentication and Key Agreement* Protokoll. Es ist für den Austausch eines Schlüssels zur weiteren Kommunikation zuständig.

Die Entitäten des 4G EPS-AKA Protokolls lassen sich in die drei Teile *User Equipment*, *Serving Network* und *Home Network* unterteilen. Das UE ist das *User Equipment* und stellt die Komponenten dar, die sich bei dem Benutzer befinden, z.B. im Smartphone. Die Evolved Node-B (eNodeB) und die Mobility Management Entity (MME) sind Teil des *Serving Network* und befinden sich bei dem Netzbetreiber, der das Netzwerk dem Benutzer zur Verfügung stellt. Der Home Subscriber Server (HSS) ist Teil des *Home Network* und befindet sich bei dem Netzbetreiber, der dem Benutzer z.B. die SIM-Karte ausgestellt hat [Cab19].

3.1.1 Ziele

Ziel des 4G EPS-AKA Protokolls ist es, wie auch beim 5G-AKA Protokoll, den Benutzer und das Netzwerk gegenseitig zu authentifizieren und einen gemeinsamen *Anchor Key* bereitzustellen der von nachfolgenden Protokollen verwendet werden kann. Im Gegensatz zum 5G-AKA Protokoll weiß das *Home Network* beim 4G EPS-AKA Protokoll jedoch nicht über den Erfolg der Authentifizierung Bescheid. Aus dem *Anchor Key* können ebenfalls weitere Schlüssel für unterschiedliche Verwendung abgeleitet werden. Eine Übersicht über die ableitbaren Schlüssel ist in Abbildung 6.2-1 und 6.2-2 des 3GPP TS 33.401 V15.4.0 zu finden [3GP18c, S. 33, 35].

3.1.2 Das Protokoll [Cab19]

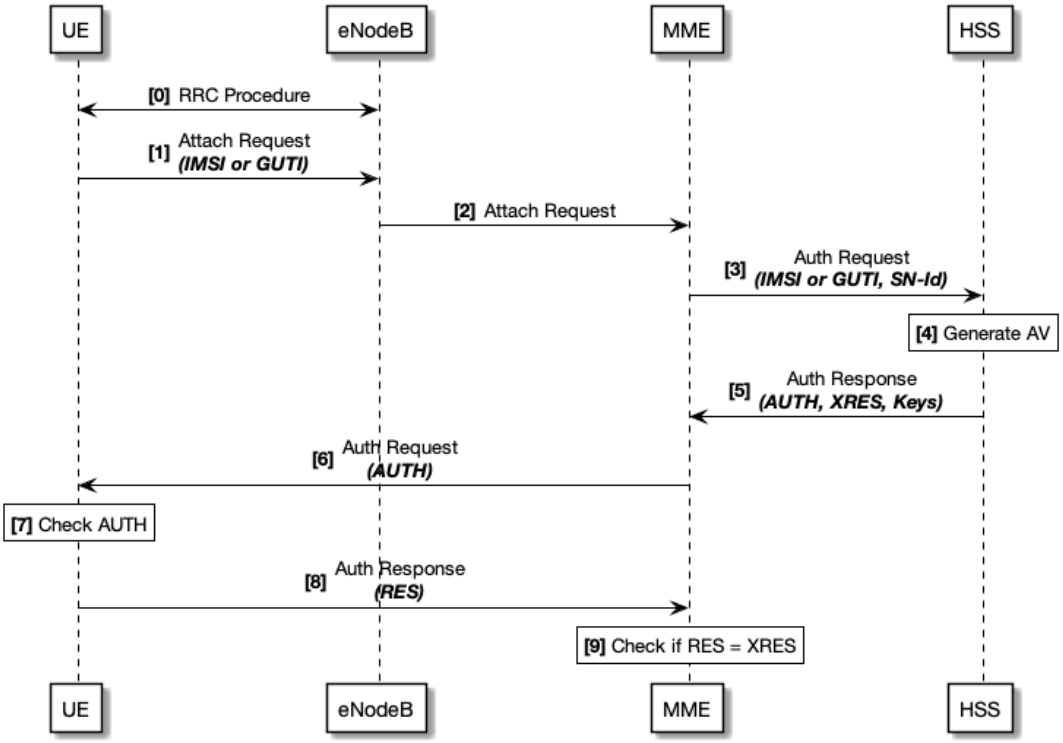

Abbildung 3.1: Das 4G EPS-AKA Protokoll

0. Bevor das 4G EPS-AKA Protokoll mit *Schritt 1* starten kann, muss die Radio Resource Control (RRC) Prozedur zwischen dem UE und der eNodeB erfolgreich abgeschlossen sein.
1. In Schritt 1 schickt das UE den *Attach Request* an den eNodeB und beginnt somit das 4G EPS-AKA Protokoll. Die Nachricht enthält entweder die IMSI oder den Globally Unique Temporary UE Identifier (GUTI) und wird durch das erfolgreiche Abschließen der RRC Prozedur ausgelöst.
2. In Schritt 2 wird der *Attach Request* aus *Schritt 1* von der eNodeB an die MME weitergesendet.
3. In Schritt 3 sendet die MME eine *Auth Request* Nachricht an den HSS. Sie enthält entweder die IMSI oder den GUTI, den die MME in *Schritt 2* von der eNodeB in der *Attach Request* Nachricht erhalten hat und zusätzlich noch die SN-Id des *Serving Network*s zu dem die MME gehört.
4. In Schritt 4 wird aus dem geheimen Schlüssel K, den sowohl der HSS wie auch das UE kennen, der Authentication Vector (AV) generiert. Er beinhaltet unter anderem die XRES, die später mit der Antwort des UE, RES, verglichen werden soll. Außerdem beinhaltet er den Authenticaton Token (AUTH), den das UE für die Berechnung der RES benötigt und kryptographische Schlüssel die im weiteren Verlauf benötigt werden.
5. In Schritt 5 wird der in *Schritt 4* generierte AV mit der *Auth Response* Nachricht an die MME gesendet.

6. In Schritt 6 sendet die MME einen *Auth Request* direkt an das UE. Er enthält den AUTH, den das UE zur Berechnung der RES benötigt.
7. Nachdem das UE den AUTH Parameter aus *Schritt 6* erhalten hat, wird er in Schritt 7 überprüft und daraus die RES berechnet.
8. In Schritt 8 wird die RES, die in *Schritt 7* berechnet wurde, in einer *Auth Response* Nachricht an die MME gesendet.
9. Nach Erhalt der *Auth Response* Nachricht wird in Schritt 9 überprüft, ob die RES, die die MME in *Schritt 8* erhalten hat, mit der XRES, die die MME in *Schritt 5* erhalten hat, übereinstimmt. Ist dies der Fall, so wird aus den kryptographischen Schlüsseln, die die MME ebenfalls in *Schritt 5* erhalten hat, unter anderem der Key for ASME (K_{ASME}) der Access Security Management Entity (ASME) berechnet. Der K_{ASME} wird beispielsweise für die Berechnung weiterer Schlüssel nach der erfolgreichen Authentifizierung benötigt [3GP18c, S. 19].

3.2 Gemeinsamkeiten von 4G EPS-AKA und 5G-AKA

Das 4G-EPS-AKA Protokoll und das 5G-AKA Protokoll haben in folgenden Aspekten Gemeinsamkeiten [Cab19]:

- Das UE hat sowohl bei 4G-EPS-AKA, wie auch bei 5G-AKA den gleichen Namen und ähnliche Aufgaben. In beiden Protokollen beginnt bei dem UE die Authentifizierung und ist der symmetrische Langzeitschlüssel K gespeichert.
- Sowohl im 4G EPS-AKA wie auch im 5G-AKA Protokoll wird für die Authentifizierung der symmetrische Langzeitschlüssel K benötigt. Er ist beim 4G EPS-AKA Protokoll im UE und im HSS gespeichert. Beim 5G-AKA Protokoll ist K im UE und im UDM/ARPF gespeichert.
- Das *Serving Network* kann, wie auch das UE, eindeutig identifiziert werden. Beim 4G EPS-AKA Protokoll wird das *Serving Network* durch die SN-Id identifiziert. Beim 5G-AKA Protokoll hingegen wird das *Serving Network* durch den SN-Name identifiziert.

  Die SN-Id und der SN-Name unterscheiden sich jedoch nur dadurch, dass dem SN-Name zusätzlich noch ”5G:” vorangestellt wurde.

3.3 Unterschiede von 4G EPS-AKA und 5G-AKA

Das 4G-EPS-AKA Protokoll und das 5G-AKA Protokoll unterscheiden sich in folgenden Aspekten [Cab19]:

- Das *Serving Network* besteht im 4G EPS-AKA Protokoll aus der MME. Die eNodeB kann auch als Teil des *Serving Network*s gesehen werden. Im 5G-AKA Protokoll hingegen besteht das *Serving Network* nur aus der SEAF.
- Im 4G EPS-AKA Protokoll besteht das *Home Network* nur aus dem HSS. Im 5G-AKA Protokoll hingegen besteht das *Home Network* aus der AUSF und aus dem UDM/ARPF/SIDF.
- Jedes UE hat ein Bitfeld, das es eindeutig identifiziert. Dieses ist jedoch bei den beiden Protokollen unterschiedlich.

  Beim 4G EPS-AKA Protokoll kann bei der Kommunikation zwischen UE und *Serving Network* die IMSI oder der GUTI und bei der Kommunikation zwischen *Serving Network* und *Home Network* nur die IMSI verwendet werden um das UE eindeutig zu identifizieren. Beim 5G-AKA Protokoll hingegen kann bei der Kommunikation zwischen UE und *Serving Network* der SUCI oder der 5G-GUTI und bei der Kommunikation zwischen *Serving Network* und *Home Network* der SUCI oder der SUPI verwendet werden um das UE eindeutig zu identifizieren.

  Beim 5G-AKA Protokoll wird zwischen UE und *Serving Network* die Identität des UE nur verschlüsselt, in Form des SUCI oder des 5G-GUTI, versendet. Beim 4G EPS-AKA Protokoll hingegen wird auch die unverschlüsselte IMSI zwischen UE und *Serving Network* versendet.
- Die Entscheidung, ob die Authentifizierung erfolgreich war, wird bei beiden Protokollen unterschiedlich gehandhabt.

  Beim 4G EPS-AKA Protokoll entscheidet nur die MME, die Teil des *Serving Network*s ist, ob eine Authentifizierung erfolgreich war. Die Entscheidung wird somit nur innerhalb des *Serving Network*s getroffen. Das *Home Network* ist nicht an der Entscheidung beteiligt und wird auch nicht über den Ausgang der Authentifizierung informiert.

  Beim 5G-AKA Protokoll hingegen entscheiden sowohl die SEAF, die Teil des *Serving Network*s ist, wie auch die AUSF, die Teil des *Home Network*s ist, ob die Authentifizierung erfolgreich war. Die Entscheidung wird somit sowohl innerhalb des *Serving Network*s, wie auch innerhalb des *Home Network*s getroffen. Des Weiteren benachrichtigt die AUSF auch das UDM über den Ausgang der Authentifizierung.

4 Beschreibung einer Sicherheitslücke

Noch bevor eine endgültige Fassung des 5G-AKA Protokolls feststand haben sich einige Forscher bereits mit unterschiedlichen Sicherheitsaspekten des Protokolls auseinandergesetzt und teilweise sogar mögliche Sicherheitslücken entdeckt. In *Security vulnerability in 5G-AKA draft* wurde solch eine Sicherheitslücke und dessen Auswirkungen, sowie mögliche Lösungsvorschläge vorgestellt, die in diesem Kapitel erläutert werden [DC18]. Ihre Befunde beziehen sich auf die Spezifikation TS 33.501 V0.7.0 des 3GPP [3GP18d]. Mittlerweile hat das 3GPP bereits neuere Versionen der Spezifikation veröffentlicht, darunter auch die Version V15.34.1, auf der die Beschreibung aus Kapitel 2 beruht [3GP19c]. Zur Verbesserung der Übersichtlichkeit wird daher im weiteren Verlauf die Sicherheitslücke im Kontext der Spezifikation TS 33.501 V15.34.1 beschrieben.

4.1 Beschreibung der Sicherheitslücke

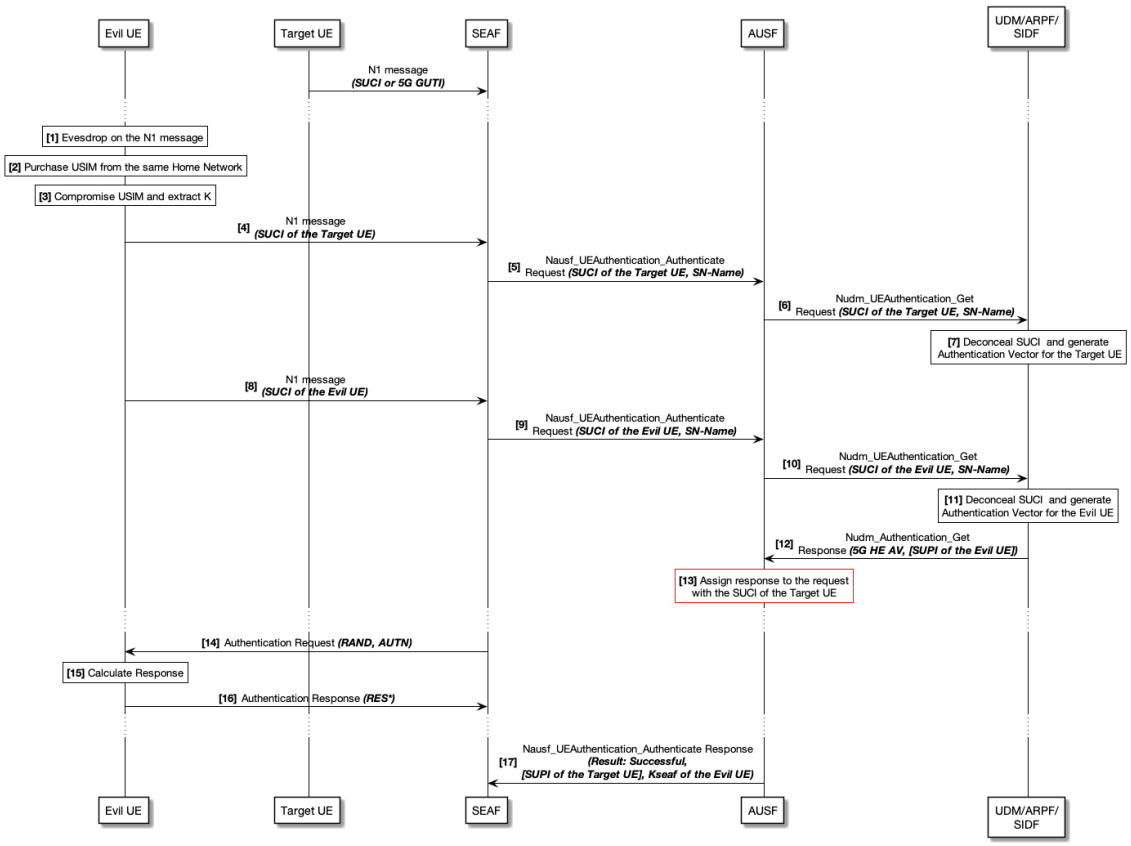

Abbildung 4.1: Beschreibung der Sicherheitslücke

Die in dem *Security vulnerability in 5G-AKA draft* vorgestellte Sicherheitslücke ermöglicht es einem Angreifer, sich als ein anderer Benutzer auszugeben [DC18]. Zur Vereinfachung wird hier davon ausgegangen, dass das UDM/ARPF/SIDF immer das 5G-AKA Protokoll auswählt und nicht das EAP-AKA' Protokoll.

Bei der vorgestellten Sicherheitslücke handelt es sich um eine *Race Condition*. Der Angriff funktioniert also nicht in allen Versuchen.

4.1.1 Vorbereitungen

1. Der Angreifer muss an den SUCI des Benutzers kommen, für den er sich ausgeben möchte. Dafür muss er die *N1 message* des Benutzers mithören und aufzeichnen.
2. Hat der Angreifer die *N1 message* des Benutzers aufgezeichnet, dann bringt er in Erfahrung zu welchem *Home Network* der SUCI gehört und kauft sich ein legitimes USIM (z.B. eine SIM-Karte) des selben *Home Network*s.
3. Ist der Angreifer im Besitz des legitimen USIM, so kompromittiert er dieses und extrahiert daraus den Langzeitschlüssel K.

4.1.2 Der Hauptteil des Angriffs

4. Der Angreifer sendet nun eine *N1 message* an die SEAF und initiiert somit eine neue Authentifikation. Diese Nachricht enthält den SUCI des Benutzers, den der Angreifer in *Schritt 1* abgehört hat.
5. Der SUCI des abgehörten Benutzers wird von der SEAF an die AUSF weitergeleitet.
6. Das UDM erhält den SUCI des abgehörten Benutzers und beginnt die erhaltene Nachricht zu verarbeiten.
7. Das UDM berechnet den SUPI des abgehörten Benutzers aus dem SUCI den es in *Schritt 6* erhalten hat und generiert den 5G HE AV für den SUCI des abgehörten Benutzers.
8. Direkt nach dem versenden der Nachricht aus *Schritt 4* sendet der Angreifer erneut eine *N1 message* jedoch dieses mal mit dem SUCI des USIMs das er in *Schritt 2* gekauft hat.
9. Auch der SUCI des Angreifers wird von der SEAF an die AUSF weitergeleitet.
10. Das UDM erhält den SUCI des Angreifers kurz nachdem es den SUCI des abgehörten Benutzers erhalten hat.
11. Das UDM berechnet den SUPI des Angreifers aus dem SUCI, den es in *Schritt 9* erhalten hat und generiert den 5G HE AV für den SUCI des Angreifers.
12. Das UDM hat in *Schritt 7 & 11* erst den SUCI des abgehörten Benutzers und dann den SUCI des Angreifers erhalten. Nun sendet es aber erst den Authentifikations-Vektor für den SUCI des Angreifers an die AUSF zurück. Das UDM sendet also die Antworten auf die erhaltenen Nachrichten nicht in der gleichen Reihenfolge zurück in der es die SUCIs erhalten hat.

In diesem Schritt (Schritt 12) wird in der hier beschriebenen Version(V15.34.1) [3GP19c] zusätzlich zu dem 5G HE AV auch teilweise der SUPI mitgesendet. Dies ist bei der Version(V0.7.0) [3GP18d], auf der in dem *Security vulnerability in 5G-AKA draft* eingegangen wird, nicht der Fall.

13. Die AUSF erhält zuerst die Antwort auf die Nachricht aus *Schritt 11*. Da die Antwort des UDM nicht den SUCI enthält, kann die AUSF nicht überprüfen zu welchem SUCI die Antwort gehört und nimmt daher fälschlicherweise an, dass die erhaltene Antwort zur Nachricht aus *Schritt 6* gehört.
14. In Schritt 14 erhält der Angreifer von der SEAF den Authentication Request. Diese Nachricht enthält RAND und AUTN die das UDM für den SUCI des Angreifers, die SEAF jedoch nimmt an, dass RAND und AUTN für die Authentifizierung des abgehörten Benutzers bestimmt sind.
15. Der Angreifer kann nun RES* aus RAND und AUTN berechnen, da er in *Schritt 3* den dafür benötigten Langzeitschlüssel K erhalten hat.
16. Der Angreifer sendet die in *Schritt 15* berechnete RES* an die SEAF.
17. Die SEAF und die AUSF überprüfen die RES* aus *Schritt 16* und akzeptieren sie schließlich. Das 5G-AKA Protokoll wurde also erfolgreich ausgeführt. Jedoch nehmen SEAF und AUSF an, dass sich der abgehörte Benutzer erfolgreich authentifiziert hat und nicht der Angreifer. Der Angreifer kann sich nun also dem SEAF gegenüber als der abgehörte Benutzer ausgeben.

    Auch in diesem Schritt (Schritt 16) wird in der hier beschriebenen Version(V15.34.1) [3GP19c] unter bestimmten Umständen der SUPI mitgesendet, nachdem er in *Schritt 12* zwischengespeichert wurde. Dies ist bei der Version(V0.7.0) [3GP18d], auf der in dem *Security vulnerability in 5G-AKA draft* eingegangen wird, nicht der Fall.

4.2 Verletzte Eigenschaft

Nachdem der Angriff erfolgreich ausgeführt wurde sind die SEAF, die AUSF und der Angreifer im Besitz des *Anchor Key*s K_{SEAF} und haben sich auf diesen geeinigt [DC18]. Außerdem glauben SEAF und die AUSF, dass der K_{SEAF} nur im Besitz des angehörten Benutzers und nicht im Besitz des Angreifers ist. Es wird also die Eigenschaft verletzt, dass der *Anchor Key* K_{SEAF} dem Angreifer gegenüber geheim sein soll.

Ein Angreifer kann somit, sofern er im Besitz eines beliebigen validen Langzeitschlüssels K ist, sich als jeden anderen Benutzer ausgeben, der bei dem gleichen Provider eine USIM gekauft hat und dessen SUCI dem Angreifer bekannt ist.

4.3 Auswirkungen der Sicherheitslücke

Diese Sicherheitslücke erlaubt es einem Angreifer sich als ein anderer Benutzer auszugeben und so Dienstleistungen des Providers auf dessen Kosten in Anspruch zu nehmen [DC18]. Der Angriff kann unabhängig von der beim SUCI verwendeten *Protection Scheme* erfolgreich ausgeführt werden, es ist also nicht notwendig dass für die Verschlüsselung des SUCI der *Null Scheme* verwendet wurde. Da der Angriff auf einer *Race Condition* beruht kann nicht garantiert werden, dass er bei jedem Durchlauf erfolgreich durchgeführt wird.

Dieser Angriff erlaubt es nicht die Nachrichten zwischen einem abgehörten Benutzer und der SEAF zu entschlüsseln.

In der hier beschriebenen Version(V15.34.1) wird in *Schritt 7 & 17* aus Abbildung 2.1 unter bestimmten Umständen auch der SUPI mitgesendet. Dies ist in der im *Security vulnerability in 5G-AKA draft* betrachteten Version(V0.7.0) nicht der Fall und könnte die Funktionalität der Sicherheitslücke beeinträchtigen.

4.4 Bekannte Lösungsvorschläge

M. Dehnel-Wild und *C. Cremers* haben in dem *Security vulnerability in 5G-AKA draft* [DC18] verschiedene Lösungsvorschläge beschrieben.

In dem ersten Lösungsansatz wird vorgeschlagen, dass in der *Nudm_Authentication_Get Response* Nachricht aus *Schritt 7* in Abbildung 2.1 zusätzlich SUCI & SUPI und dass in der *Nausf_UEAuthentication_Authenticate Response* Nachricht aus *Schritt 10* aus Abbildung 2.1 zusätzlich der SUCI mitgesendet wird. Dies ermöglicht es der AUSF und der SEAF die erhaltenen Nachrichten eindeutig zuzuordnen und somit einen erfolgreichen Angriff zu verhindern.

Ein anderer Lösungsansatz schlägt vor, dass die Nachrichten zwischen der AUSF und dem UDM eine eindeutige Zufallszahl enthalten und somit klar zugeordnet werden können.

Eine andere Möglichkeit besteht darin, durch die Implementierung und andere technische Maßnahmen eine erfolgreiche Durchführung des Angriffs zu verhindern und somit sicherstellen, dass sich ein Angreifer nicht als einen anderen Benutzer ausgibt. Dieser Lösungsansatz wird jedoch nicht empfohlen, da diese Maßnahmen jeder Provider separat vornehmen müsste und die Sicherheitslücke rein theoretisch trotzdem noch möglich ist.

5 Implementierung der Sicherheitslücke

Die in Kapitel 4 beschriebene Sicherheitslücke und das 5G-AKA Protokoll wurde nun implementiert. Die Implementierung ist dieser Arbeit beigefügt und kann auch auf Github unter folgendem Link eingesehen werden: https://github.com/rbroesamle/5G-AKA-Protocol/tree/master/Implementation In diesem Kapitel wird sich auf den Commit mit der ID 95f76c00aff8f462554ee8b277f747eda251e3d6 bezogen.

Es wurde das 5G-AKA Protokoll aus der Spezifikation TS 33.501 des 3GPP mit der Version V15.34.1 implementiert [3GP19c]. Die Sicherheitslücke wurde darauf aufbauend implementiert.

5.1 Ausführen des Programms

Zum Ausführen des Programms wurde ein *Gradle* Build Script bereitgestellt. Es wird daher empfohlen die neueste Java und Gradle Version auf einem Linux Betriebssystem installiert zu haben.

Um das Programm auszuführen muss der Befehl `gradle run` in eine Konsole eingegeben werden. Dafür sollte man sich in dem Ordner befinden, in dem sich auch die `build.gradle` Datei befindet.

Alternativ kann auch der Gradle Wrapper ausgeführt werden. Dafür muss auf Linux oder macOS der Befehl `./gradlew run` und auf Windows der Befehl `gradlew.bat run` ausgeführt werden. Auch hier sollte man sich in dem Ordner befinden, in dem sich auch die build.gradle Datei befindet.

5.1.1 Einstellungen

Das Programm erlaubt es Einstellungen anzupassen, die das Verhalten des Programms beeinflussen. Diese Einstellungen sind in der `Implementation.App` Klasse zu finden und lassen sich folgendermaßen beschreiben:

RUN_MODE

Der *RUN_MODE* beschreibt ob das Protokoll oder die Sicherheitslücke simuliert werden soll. Er kann die Werte `RunMode.Protocol` und `RunMode.Vulnerability` annehmen.

Wird der *RUN_MODE* auf den Wert `RunMode.Protocol` gesetzt, dann wird ein erfolgreicher normaler Durchlauf des Protokolls simuliert. Nach dem Durchlauf des Protokolls wird ausgegeben ob der Durchlauf erfolgreich war.

Wird der *RUN_MODE* auf den Wert `RunMode.Vulnerability` gesetzt, dann wird der Angriff, wie er in Kapitel 4 beschrieben ist, simuliert. Nach dem Durchlauf wird ausgegeben ob Authentifizierung und Angriff erfolgreich waren. Der Angriff kann nur erfolgreich sein wenn der Langzeitschlüssel K_{SEAF} bei dem UE und der SEAF Entitäten gleich ist. Mit der *DETAILED_AUTH_INFO* Einstellung lassen sich die Langzeitschüssel auf der Konsole ausgeben. Da der Angriff auf einer *Race Condition* beruht ist auch das Fehlschlagen des simulierten Angriffs zu erwarten.

LOG_MESSAGES

Mit *LOG_MESSAGES* lässt sich einstellen ob die von Entitäten empfangenen Nachrichten auf der Konsole ausgegeben werden sollen. Es können die Werte `true` und `false` angegeben werden.

Wird *LOG_MESSAGES* auf `true` gesetzt, dann werden auf der Konsole alle Nachrichten ausgeben, die von einer der Entitäten empfangen wurden. Die Ausgabe auf der Konsole hat das Format: *SENDER -> EMPFÄNGER : NACHRICHT*

Wird *LOG_MESSAGES* auf `false` gesetzt, dann werden die empfangenen Nachrichten nicht auf der Konsole ausgegeben.

DETAILED_AUTH_INFO

Mit *DETAILED_AUTH_INFO* lässt sich einstellen ob detaillierte Informationen auf der Konsole ausgegeben werden sollen. Es können die Werte `true` und `false` angegeben werden.

Wird *DETAILED_AUTH_INFO* auf `true` gesetzt, werden detaillierte Informationen zur Authentifizierung auf der Konsole ausgegeben. Dies beinhaltet Informationen darüber ob die Authentifizierung von der SEAF, der AUSF und dem UDM als erfolgreich angesehen wird, wie es in *Schritt 14 & 16* von Abbildung 2.1 beschrieben ist. Zusätzlich wird der Langzeitschlüssel K_{SEAF} der im UE und in der SEAF gespeichert ist ausgegeben.

Wird *DETAILED_AUTH_INFO* auf `false` gesetzt, werden keine detaillierten Informationen zur Authentifizierung auf der Konsole ausgegeben.

5.2 Beschreibung des Aufbaus

Für die Implementierung wurde die Programmiersprache Java gewählt.

Der Einstiegspunkt des Programms ist die `Implementation.App` Klasse. Von dort lässt sich das Programm mit der `main` Methode aufrufen.

In dem `Implementation.structure` Package wird die grundlegende Struktur für Entitäten und Nachrichten definiert. Alle Entitäten haben die `Implementation.structure.Entity` Klasse als Superklasse und alle Nachrichten haben das `Implementation.structure.Message` Interface implementiert.

Das `Implementation.helper` Package beinhaltet Helferklassen. Diese Helferklassen werden verwendet um die Übersichtlichkeit in den anderen Packages zu verbessern. Sie beinhalten nur statische Methoden.

In dem `Implementation.protocol` Package ist die tatsächliche Implementierung des Protokolls zu finden. Es ist unterteilt in die Packages `Implementation.protocol.additional`, `Implementation.protocol.data`, `Implementation.protocol.entities` und `Implementation.protocol.messages`.

Das `Implementation.protocol.additional` Package beinhaltet hauptsächlich Klassen die kryptographische Berechnungen ausführen, wie die KDF. Alle Klassen in diesem Package beinhalten nur statische Methoden.

Das `Implementation.protocol.data` Package beinhaltet Klassen, die mehrere verschiedene Parameter in sich zusammenfassen, wie z.B. der 5G HE AV.

In dem `Implementation.protocol.entities` Package ist die Implementierung der vier Entitäten UE, SEAF, AUSF und UDM zu finden, so wie eine `EvilUE` Klasse, die den Angreifer darstellen soll. In diesen Klassen ist das Versenden und Antworten auf Nachrichten implementiert. Dies geschieht über die `Implementation.structure.Entity` Superklasse.

In dem `Implementation.protocol.messages` Package sind alle Nachrichten zu finden, die die Entitäten untereinander verschicken. Jede Nachricht hat eine eigene Klasse, die das `Implementation.structure.Message` Interface implementiert.

5.2.1 Implementierung der Race Condition

Um die *Race Condition* zu realisieren muss sichergestellt werden, dass versendete Nachrichten sich überholen können. Dies wurde mit Hilfe von *Threads* implementiert. Dafür wurden in der `Implementation.structure.Entity` Klasse die `sendMessage` Methode erstellt. Da die `Implementation.structure.Entity` Klasse die Superklasse von allen Entitäten ist wird in den Entitäten die `sendMessage` der `Implementation.structure.Entity` Klasse verwendet um Nachrichten zu versenden.

Führt eine Entität die `sendMessage` Methode aus, so wird ein neuer Thread erzeugt der die `onReceiveMessage` Methode asynchron auf der Entität ausführt, die die gesendete Nachricht erhalten soll. Somit ist sichergestellt, dass eine *Race Condition* möglich ist.

5.2.2 Unterschiede zum Protokoll

Ziel dieser Implementierung ist es die Praktikabilität der in Kapitel 4 beschriebenen Sicherheitslücke herauszufinden. In der Implementierung wurden daher einige Teile der Protokolls, die für die Sicherheitslücke nicht relevant sind, verändert oder weggelassen. Nachfolgend ist eine Liste aller Abweichungen vom Protokoll aufgeführt. Alle Abweichungen sind im Code mit dem Kommentar `//MARK: Deviation #` versehen. Das `#` entspricht der Nummer der Abweichung.

1. In der `Implementation.protocol.additional.AVGenerator` Klasse wurde bei der Generierung der SQN in der `generateSQN` Methode von der Spezifikation abgewichen. In der Spezifikation wurden Bedingungen, wie die *Wrap around Protection*, für die Generierung der SQN festgelegt. In der Implementierung wurde jedoch die SQN mit einem Zufallsgenerator generiert, der diese Bedingungen nicht erfüllt.

2. In der `Implementation.protocol.additional.KGF` Klasse wurde bei der *Message authentication function* f1 von der Spezifikation abgewichen. In der Spezifikation ist für die `f1` Methode eine *Message authentication function* gefordert. In der Implementierung wurde hierfür jedoch die letzten 8 Bytes des *HmacSHA256* verwendet.

3. In der `Implementation.protocol.additional.KGF` Klasse wurde bei der *Message authentication function* f2 von der Spezifikation abgewichen. In der Spezifikation ist für die `f2` Methode eine *Message authentication function* gefordert. In der Implementierung wurde hierfür jedoch der *HmacSHA256* verwendet.

4. In der `Implementation.protocol.additional.MAF` Klasse wurde bei der *Key generation function* f3 von der Spezifikation abgewichen. In der Spezifikation ist für die `f3` Methode eine *Key generating function* gefordert. In der Implementierung wurde hierfür jedoch die letzten 16 Bytes des *HmacSHA256* verwendet.

5. In der `Implementation.protocol.additional.MAF` Klasse wurde bei der *Key generation function* f4 von der Spezifikation abgewichen. In der Spezifikation ist für die `f4` Methode eine *Key generating function* gefordert. In der Implementierung wurde hierfür, wie auch bei der Abweichung Nr. 4, die letzten 16 Bytes des *HmacSHA256* verwendet.

6. In der `Implementation.protocol.additional.MAF` Klasse wurde bei der *Key generation function* f5 von der Spezifikation abgewichen. In der Spezifikation ist für die `f5` Methode eine *Key generating function* gefordert. In der Implementierung wurde hierfür jedoch die letzten 6 Bytes des *HmacSHA256* verwendet.

7. In der `Implementation.protocol.additional.SIDF` Klasse wurde bei der Entschlüsselung des SUCI in den SUPI von der Spezifikation abgewichen. In der Spezifikation ist der SUCI ein kompliziertes Bitfeld und bedarf somit einer umfangreichen Entschlüsselungsfunktion. Die Implementierung der `deconcealSUCI` Methode verwendet jedoch nur das *RSA* Verfahren zum entschlüsseln.

8. In der `Implementation.protocol.additional.SIDF` Klasse wurde bei der Verschlüsselung des SUPI in den SUCI von der Spezifikation abgewichen. In der Spezifikation ist der SUCI ein kompliziertes Bitfeld. Dies muss eine Verschlüsselungsfunktion berücksichtigen. Die Implementierung der `deconcealSUCI` Methode verwendet jedoch nur das *RSA* Verfahren zum verschlüsseln. Daher entspricht der SUCI in der Implementierung auch nur dem Ergebnis des *RSA* Verfahrens und nicht dem Bitfeld das in der Spezifikation beschrieben wurde.

9. Da in der Sicherheitslücke der 5G-GUTI nicht verwendet wird, wurde der 5G-GUTI nicht implementiert.

10. In *Schritt 3* der Abbildung 2.1 soll die AUSF einer SEAF, die den mitgesendeten SN-Name nicht verwenden darf, mit einer *serving network not authorized* Nachricht antworten. In der Implementierung sendet die `Implementierung.protocol.entities.AUSF` Klasse diese Nachricht nicht.

11. In *Schritt 3* der Abbildung 2.1 soll die AUSF überprüfen ob eine SEAF, die eine Nachricht an die AUSF geschickt hat, den mitgesendeten SN-Name verwenden darf. In der Implementierung erfolgt dies von der `Implementation.protocol.entities.AUSF` Klasse nicht.

12. In *Schritt 17* der Abbildung 2.1 wird beschrieben, dass die AUSF der SEAF den SUPI des Benutzer mitsendet, wenn die AUSF diesen in *Schritt 7* erhalten hat. Hat sie den SUPI nicht in *Schritt 7* erhalten, so wurde in *Schritt 1* der 5G-GUTI an die SEAF gesendet. In der `Implementation.protocol.entities.SEAF` Klasse wurde die Nachricht aus *Schritt 17* jedoch nur mit mitgesendetem SUPI behandelt.
13. Erhält die SEAF eine *Authentication Failure* Nachricht vom UE so soll sie im Falle einer *Synchronization failure* eine neue Authentifizierung initiieren können. In der Implementierung wird in der `Implemenation.protocol.entities.SEAF` Klasse jedoch nie eine neue Authentifizierung initiiert.
14. In *Schritt 14* der Abbildung 2.1 soll die SEAF bei einer zu spät oder nicht erhaltenen Nachricht die Authentifizierung abbrechen. In der Implementierung wird die erhaltene Nachricht jedoch immer akzeptiert. Es gibt also keinen *Timeout*.
15. In *Schritt 12* der Abbildung 2.1 wird überprüft ob der AUTN angenommen werden kann. In der Implementierung wurde in der `Implementation.protocol.entities.UE` Klasse diese Überprüfung weggelassen.
16. In *Schritt 2** der Abbildung 2.2 sendet das UE eine *Authentication Failure* Nachricht an die SEAF. Diese Nachricht enthält auch einen CAUSE Wert, der den Grund des Fehlschlagens beschreibt. In der Implementierung wird jedoch kein CAUSE Wert in der *Authentication Failure* Nachricht mitgesendet.
17. In *Schritt 11* der Abbildung 2.1 sendet die SEAF in der *Authentication Request* Nachricht zusätzlich noch den ngKSI und die ABBA mit. In der Implementierung werden diese Parameter jedoch nicht mitgesendet.
18. In *Schritt 15* der Abbildung 2.1 sendet die SEAF eine *Nausf_UEAuthentication_Authenticate Request* Nachricht an die AUSF. In der Implementierung wurde diese Nachricht in *Nausf_UEAuthentication_Confirmation Request* umbenannt, da die Nachricht aus *Schritt 2* sonst den gleichen Name hätte.
19. In *Schritt 17* der Abbildung 2.1 sendet die AUSF eine *Nausf_UEAuthentication_Authenticate Response* Nachricht an die SEAF. In der Implementierung wurde diese Nachricht in *Nausf_UEAuthentication_Confirmation Response* umbenannt, da die Nachricht aus *Schritt 10* sonst den gleichen Name hätte.
20. In *Schritt 7* der Abbildung 2.1 wird zusätzlich zum 5G HE AV der Hinweis mitgesendet, dass der 5G HE AV für das 5G-AKA Protokoll bestimmt ist. In der Implementierung wurde dieser Hinweis weggelassen.
21. Der SUPI ist als ein Bitfeld spezifiziert, das aus mehreren Parametern besteht. In der Implementierung ist der SUPI jedoch nur eine zufällige Bitfolge.

5.3 Übersicht der Komponenten

In Tabelle 5.1 wurden alle Begriffe aufgelistet, die als Klassen in der Implementierung zu finden sind. In Tabelle 5.2 wurden alle Begriffe aufgelistet, die in der Implementierung erzeugt werden. In den dazugehörigen Klassen werden die Begriffe erzeugt.

Begriff	Klasse
5G-GUTI	Implementation.protocol.data.Data_5G_GUTI
5G HE AV	Implementation.protocol.data.Data_5G_HE_AV
5G SE AV	Implementation.protocol.data.Data_5G_SE_AV
AUTN	Implementation.protocol.data.Data_AUTN
f1, f2, f3, f4, f5	Implementation.protocol.additional.MAF, .KGF
KDF	Implementation.protocol.additional.KDF
UE	Implementation.protocol.entities.UE
SEAF	Implementation.protocol.entities.SEAF
AUSF	Implementation.protocol.entities.AUSF
UDM	Implementation.protocol.entities.UDM
SIDF	Implementation.protocol.additional.SIDF

Tabelle 5.1: Übersicht der normalen Klassen

Begriff	Klasse
AK	Implementation.protocol.additional.AVGenerator, Implementation.protocol.entities.UE
AMF	Implementation.App
CK	Implementation.protocol.additional.AVGenerator, Implementation.protocol.entities.UE
HRES*	Implementation.protocol.entities.SEAF
HXRES*	Implementation.protocol.entities.AUSF
IK	Implementation.protocol.additional.AVGenerator, Implementation.protocol.entities.UE
K	Implementation.App
K_{AUSF}	Implementation.protocol.entities.UDM, Implementation.protocol.entities.UE
K_{SEAF}	Implementation.protocol.entities.AUSF, Implementation.protocol.entities.UE
MAC	Implementation.protocol.additional.AVGenerator
RAND	Implementation.protocol.additional.AVGenerator
RES	Implementation.protocol.entities.UE
RES*	Implementation.protocol.entities.UE
SN-Name	Implementation.App
SQN	Implementation.protocol.additional.AVGenerator
SUCI	Implementation.protocol.additional.SIDF
SUPI	Implementation.App
XMAC	Implementation.protocol.entities.UE
XRES	Implementation.protocol.additional.AVGenerator
XRES*	Implementation.protocol.entities.UDM

Tabelle 5.2: Übersicht der erzeugenden Klassen

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde das 5G-AKA Protokoll erklärt und mit dem 4G EPS-AKA Protokoll verglichen. Außerdem wurde eine Sicherheitslücke des 5G-AKA Protokolls erklärt und implementiert.

Das 5G-AKA Protokoll ist eines von zwei Protokollen des 5G-Standards das für den sicheren Austausch eines kryptographischen Schlüssels zuständig ist. Es besteht aus den 4 Entitäten UE, SEAF, AUSF und UDM. Das UE entspricht z.B. dem Smartphone eines Benutzers, die SEAF entspricht z.B. einem Funkmast mit dem sich das Smartphone des Benutzers verbindet und die AUSF und das UDM entsprechen z.B. dem Provider, von dem der Benutzer eine SIM-Karte gekauft hat. Über diese vier Entitäten werden nun Nachrichten versendet, die das UE authentifizieren und einen gemeinsamen Schlüssel austauschen. War das 5G-AKA Protokoll erfolgreich, dann verfügen das UE und die SEAF über einen *Anchor Key*, genannt K_{SEAF}, der für die weitere Verschlüsselung verwendet wird.

Das 5G-AKA Protokoll hat einige Verbesserungen gegenüber dem 4G EPS-AKA Protokoll des 4G-Standards. Jedoch wurde auch in dem 5G-AKA Protokoll eine Sicherheitslücke entdeckt. In dieser Sicherheitslücke geht es darum, dass sich ein Angreifer als ein anderer Benutzer ausgeben kann. Der Angreifer muss dafür den verschlüsselten Identifier des Benutzers, genannt SUCI, abhören, sowie ein SIM-Karte bei dem Provider kaufen, bei dem auch der Benutzer seine SIM-Karte gekauft hat. Der Angreifer hat nun die Möglichkeit sich als diesen Benutzer auszugeben. Der Angriff beruht auf einer *Race Condition*, ist also nicht immer erfolgreich. Des Weiteren wurde der Angriff für eine ältere Version des 5G-AKA Protokolls gefunden. Es kann also sein, dass dieser Angriff bis zur kommerziellen Verwendung des 5G-Standards nicht mehr möglich ist.

Ausblick

Das 5G-AKA Protokoll befindet sich zurzeit noch in der Entwicklung und ist bisher nur in wenigen Städten verfügbar. Während der Erstellung dieser Arbeit wurden auch bereits mehrere neue Versionen der Spezifikation des 5G-AKA Protokolls veröffentlicht. Diese Entwicklung zeigt, dass sich das Protokoll bis zu seiner flächendeckenden Verwendung noch verändern und weiterentwickeln wird. Daher ist es nahelegend, dass die hier vorgestellte Sicherheitslücke einen Benutzer möglicherweise nicht betreffen wird.

Literaturverzeichnis

[3GP18a] 3GPP TS 33.102 V15.1.0. „3G Security“. In: (Dez. 2018). URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2262 (zitiert auf S. 20–22, 25–29).

[3GP18b] 3GPP TS 33.220 V15.4.0. „Generic Authentication Architecture (GAA)“. In: (Dez. 2018). URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2280 (zitiert auf S. 27).

[3GP18c] 3GPP TS 33.401 V15.4.0. „3GPP System Architecture Evolution (SAE)“. In: (Juni 2018). URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2296 (zitiert auf S. 33, 35).

[3GP18d] 3GPP TS 33.501 V0.7.0. „Security architecture and procedures for 5G system“. In: (Feb. 2018). URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169 (zitiert auf S. 37, 39).

[3GP19a] 3GPP TS 23.003 V15.7.0. „Numbering, addressing and identification“. In: (Juni 2019). URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=729 (zitiert auf S. 20, 23, 24, 26, 28, 29).

[3GP19b] 3GPP TS 24.501 V16.1.0. „Non-Access-Stratum (NAS) protocol for 5G System (5GS)“. In: (Juni 2019). URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3370 (zitiert auf S. 30).

[3GP19c] 3GPP TS 33.501 V15.34.1 (15.4.0). „Security architecture and procedures for 5G system“. In: (März 2019). URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169 (zitiert auf S. 15, 16, 18, 20–24, 27–31, 37, 39, 41).

[Bro17] G. Brown. „Service-Based Architecture for 5G Core Networks“. In: *A Heavy Reading white paper produced for Huawei Technologies Co. Ltd. Online verfügbar unter: https://www. huawei. com/en/press-events/news/2017/11/HeavyReading-WhitePaper-5G-Core-Network, letzter Zugriff am* 1 (2017), S. 2018. URL: https://www.3g4g.co.uk/5G/5Gtech_6004_2017_11_Service-Based-Architecture-for-5G-Core-Networks_HR_Huawei.pdf (zitiert auf S. 13).

[Cab19] CableLabs. „A Comparative Introduction to 4G and 5G Authentication“. In: (2019). URL: https://www.cablelabs.com/insights/a-comparative-introduction-to-4g-and-5g-authentication (zitiert auf S. 13, 33–36).

[DC18] M. Dehnel-Wild, C. Cremers. „Security vulnerability in 5G-AKA draft“. In: *Department of Computer Science, University of Oxford, Tech. Rep* (2018). URL: https://www.cs.ox.ac.uk/5G-analysis/5G-AKA-draft-vulnerability.pdf (zitiert auf S. 14, 37–40).

[Int15] Internet Engineering Task Force (IETF). „The Network Access Identifier“. In: (Mai 2015). URL: https://datatracker.ietf.org/doc/rfc7542/ (zitiert auf S. 23, 24).

[ITU08] ITU-R. „Requirements related to technical performance for IMT-Advanced radio interface (s)“. In: *International Telecommunications Union* (2008). URL: https://www.itu.int/pub/R-REP-M.2134-2008/en (zitiert auf S. 13).

[KQAW09] A. H. Khan, M. A. Qadeer, J. A. Ansari, S. Waheed. „4G as a Next Generation Wireless Network“. In: *2009 International Conference on Future Computer and Communication*. Apr. 2009, S. 334–338. DOI: 10.1109/ICFCC.2009.108. URL: https://ieeexplore.ieee.org/abstract/document/5189800 (zitiert auf S. 13).

[Kum11] K. Kumaravel. „Comparative study of 3G and 4G in mobile technology“. In: *International Journal of Computer Science Issues (IJCSI)* 8.5 (2011), S. 256. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.642.2509&rep=rep1&type=pdf#page=266 (zitiert auf S. 13).

[PDFF17] S. Parkvall, E. Dahlman, A. Furuskar, M. Frenne. „NR: The New 5G Radio Access Technology“. In: *IEEE Communications Standards Magazine* 1.4 (Dez. 2017), S. 24–30. DOI: 10.1109/MCOMSTD.2017.1700042. URL: https://ieeexplore.ieee.org/abstract/document/8258595 (zitiert auf S. 13).

[Sut18] A. Sutton. „5G network architecture“. In: *J. Inst. Telecommun. Professionals* 12.1 (2018), S. 9–15 (zitiert auf S. 13).

[WKA+12] M. Winter, S. Kunze, E. P. Adeva, B. Mennenga, E. Matûs, G. Fettweis, H. Eisenreich, G. Ellguth, S. Höppner, S. Scholze, R. Schüffny, T. Kobori. „A 335Mb/s 3.9mm2 65nm CMOS flexible MIMO detection-decoding engine achieving 4G wireless data rates“. In: *2012 IEEE International Solid-State Circuits Conference*. Feb. 2012, S. 216–218. DOI: 10.1109/ISSCC.2012.6176981. URL: https://ieeexplore.ieee.org/abstract/document/6176981 (zitiert auf S. 13).

Alle URLs wurden zuletzt am 29. 10. 2019 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift